Efficient Clustering and Visibility Calculation for Global Illumination

Daniel Meneveaux *
IRCOM-SIC, Poitiers, France

Kadi Bouatouch †
IRISA, Rennes, France

Gilles Subrenat, Philippe Blasi ‡
IRCOM-SIC, Poitiers, France

Abstract

Using a radiosity method to estimate light inter-reflections within large scenes still remains a difficult task. The two main reasons are: (i) the computations entailed by the radiosity method are time consuming and (ii) the large amount of memory needed is very large. In this paper, we address this problem by proposing a new clustering technique as well as a new method of visibility computation for complex indoor scenes. Our clustering algorithm groups polygons that are close to each other in each room (or corridor) of the building. It relies on a classification method of k-mean type and allows the use of several kinds of distance functions. For each group of polygons (or cluster), we estimate the set of potentially visible clusters with the help of openings such as doors or windows. This computation results in a graph in which the nodes correspond to clusters and the edges express visibility relationships between the corresponding clusters. We use this graph for computing radiosity in complex buildings while reducing both the amount of memory needed and the computing time. Our global illumination method is a MWRA (multi-wavelet radiosity algorithm). Unlike cluster-based radiosity methods, our MWRA does not approximate (but computes accurately) the light energy impinging or leaving a cluster after multiple reflections. We provide results for 3 different test scenes containing a high number of polygons.

CR Categories: I.3.7 [Compuer Graphics]: Three Dimensional Graphics and Realism—Radiosity;

Keywords: architectural environments, clustering, visibility, global illumination

1 Introduction

For a lot of reasons, the global illumination process (whatever the technique employed), has the particularity to be very expensive in terms of both memory resources and computing time. However, it is possible under certain assumptions to improve the performances of the algorithm. As an example, a viewer located in a building containing hundreds or thousands of rooms does not see all the surfaces of the environment. The reason is that some major occluders such as walls reduce the number of objects that can be seen from a given viewpoint. For the same reason, during lighting simulation, a surface located in a particular room gets light flux *only* from (i) the surfaces that are in the same room and (ii) those visible through portals (say, openings such as doors, windows, etc.). This set of surfaces is commonly called PVS (Potentially Visible Set).

To accelerate the global illumination process, the scene is usually subdivided into groups of surfaces, a group being a set of surfaces that are close to each other. There exists two methods for creating these groups.

The first one consists in partitioning the scene into 3D cells containing portals used to determine a graph expressing visibility relationships between the 3D cells. Several techniques have been already devised for partitioning a scene. In [Teller 1992b; T. Funkhouser and Khorramabadi 1996; John M. Airey 1990], a BSP technique is used, providing 3D cells that can be rooms, portions of rooms, corridors, portion of corridors, etc. While in [Meneveaux et al. 1998b], as the partitioning method used fits well with the scene topology, the cells are exactly rooms or corridors.

The second method creates a hierarchy of clusters of surfaces. The construction of this hierarchy relies on the use of bounding volumes [Goldsmith and Salmon 1987; Haber et al. 2000]. For more efficiency, 3D cells and clusters can be used together.

There exists two ways of using clusters in hierarchical radiosity. The first one consists in approximating the light energy impinging or leaving a cluster [Sillion 1995; Sillion and Drettakis 1995; Sillion 1994; Smits et al. 1994] for accelerating radiosity to the detriment of accuracy. As for the second, it computes the PVS of each cluster and a Cluster Visibility Graph (CVG) in a preprocessing step. This CVG makes easier visibility calculations when computing global illumination with hierarchical radiosity [T. Funkhouser and Khorramabadi 1996; Teller et al. 1994; Funkhouser 1996]. In this paper, we use this second approach. As will be shown in section 7, using PVS for clusters rather than for 3D cells reduces both the computing time and the amount of memory needed.

This paper contains two contributions in the context of hierarchical radiosity using a CVG for the reason explained above. The first one is a new technique for constructing clusters while the second is a fast and simple algorithm for estimating the PVS associated with each cluster.

This paper is organized as follows. Section 2 provides an overview of our system. Prior works concerning clustering techniques and visibility processing are given in section 3. Concerning our contribution, section 4 gives details about our clustering algorithm while our visibility computations are explained in section 5. We then discuss the radiosity algorithm (section 6) and give some results (section 7) before we conclude (section 8).

2 Overview

Our global illumination method is multi-wavelet radiosity operating on a scene partitioned into 3D cells (from now on, a cell represents a room, a corridor, etc.) according to [Meneveaux et al. 1998b]. Within each cell, we construct a set of clusters using a new method described later in the paper. A cluster visibility graph (CVG) is computed. It gives the PVS of each cluster so that for a cluster C

*daniel@sic.sp2mi.univ-poitiers.fr
†kadi@irisa.fr
‡subrenat@sic.sp2mi.univ-poitiers.fr, blasi@sic.sp2mi.univ-poitiers.fr

within a cell R, the PVS is the set of the clusters within R and those visible to C through portals. The PVS are computed with a new method presented in a following section and the solution of hierarchical radiosity is determined according to the method described in [Teller et al. 1994; Meneveaux et al. 1998a]. It consists, for each cluster, in shooting (or gathering) the energy to (or coming from) other clusters. More precisely, in a shooting approach, when a cluster S shoots its energy to a cluster R, every surface of S shoots its energy to every surface of S. The approximations methods described in [Sillion 1995; Smits et al. 1994] are not used in our algorithm.

3 Related works

The radiosity method has now become a standard for estimating light inter-reflections within 3D environments. Since the initial work of Goral [Goral et al. 1984], one of the major improvements that have been proposed for reducing the computing time is the hierarchical radiosity principle [Bouatouch and Pattanaik 1995; Gibson and Hubbold 1996; Gortler et al. 1993; Hanrahan et al. 1991; Sillion 1994; Sillion and Drettakis 1995; Smits et al. 1994]. For some of these methods, every surface of the scene is subdivided into a hierarchy of patches while the others make also use of a cluster hierarchy for speeding up light exchanges between clusters. In this section, we outline some techniques dealing with clustering, visibility processing and cluster-based hierarchical radiosity.

3.1 Clustering

Several techniques ([Hasenfratz et al. 1999]) are commonly employed for grouping polygons or objects within an environment. They are mostly used as a precomputation and provide a method for fast cluster visibility culling. We classify them into the following families :

- **Spatial subdivision** allows the scene to be subdivided into voxels, which makes faster the traversal of the scene by a ray [Amanatides and Woo 1987; Klimaszewski 1997; Cleary and Wyvill 1988; Salam et al. 1999].
- **Hierarchy of bounding volumes** such as those described in [Goldsmith and Salmon 1987; Haber et al. 2000], BSP trees [John M. Airey 1990; Airey 1990; Teller et al. 1994; Teller and Hanrahan 1993; Kay and Kajiya 1986; Haines and Wallace 1991], octrees and quadtrees allow a multi-level representation of the environment. They can be used for clustering and for speeding up ray-tracing.
- **Hybrid methods** combine hierarchy, regular grid, non-regular grid and bounding volumes in order to benefit from the advantages of the above approaches (see [Cazals et al. 1995; Snyder and Barr 1987; Gigante 1990; Müller et al. 2000]).

Basically, the criteria for constructing a hierarchy are specifically based on the scene geometry. In most cases, the number of objects contained in each node (or leaf) of the hierarchy is fixed by the user. The result is a set of clusters containing almost the same number of objects. However if this number is fixed a priori to N, when $2N$ objects are close, it is impossible to automatically create only one unique group. Moreover, if a group contains $N-1$ objects, the algorithm may insert a N^{th} far-off object yielding a cluster with a large empty space.

In [Haber et al. 2000], Haber et al. propose new subdivision criteria for creating a cluster hierarchy which takes the geometry into account while constructing different size clusters. This method also makes use of a post-processing for reorganizing the hierarchy. It can be used in several types of global illumination for constructing clusters composed of objects having the same photometric parameters, close to each other, etc.

The clustering we propose does not need any hierarchy construction nor post-processing for solving the problems given above. It takes into account the geometry and automatically creates clusters of different size, composed of objects close to each other. Our method can also consider other distance functions based on photometric criteria for example. The number of objects per cluster is not rigidly fixed and the number of clusters is automatically adapted. The user chooses an average number of objects per cluster, but the number of objects in each cluster can be much greater or much lower than this average number. This only depends on the position of the polygons within the same cluster. In this way, the empty space in the clusters is reduced.

3.2 Visibility processing

A lot of works have been proposed in order to reduce visibility computation. Basically, they rely on some preprocessing, based on the environment geometry [Coorg and Teller 1997; Teller and Hanrahan 1993; Teller 1992a; Drettakis and Puech 1997; Drettakis and Fiume 1994; Chin and Feiner 1989; Haines and Wallace 1991]. Generally speaking, this kind of work proposes a means for estimating a very precise visibility information between sets of objects (polygons, clusters of polygons, etc.) and needs complex datastructures such as tubes, blockers, etc. Another approach, proposed by Leblanc et al. [Leblanc and Poulin 2000] builds a face hierarchy for reducing visibility computation. In this paper we propose a method well suited for indoor environments allowing less precise but very fast and easy visibility computations.

3.3 Global illumination for Complex scenes

3.3.1 Hierarchical Radiosity

For complex scenes, cluster-based hierarchical radiosity has been commonly employed. This method [Smits et al. 1994; Sillion and Drettakis 1995; Sillion 1994; Sillion 1995; Gibson and Hubbold 1996] makes use of a cluster hierarchy to accelerate radiosity by approximating the light energy impinging or leaving a cluster to the detriment of accuracy.

3.3.2 Indoor Scenes

J.M. Airey was the first who applied a partitioning algorithm to complex architectural environments to represent them by a hierarchical datastructure [John M. Airey 1990], say a BSP tree. The result is a set of 3D cells (rooms corridors, etc.), containing openings (portals) used to compute a visibility graph (PVS). Teller et. al. proposed improvements on visibility computation as well as some extensions of Airey's work [Teller et al. 1994] used in hierarchical radiosity. Meneveaux et al. [Meneveaux et al. 1998b] proposed a method that precisely extracts each room of a building and determines their portals and the corresponding visibility graph [Meneveaux et al. 1998a; Meneveaux and Bouatouch 1999].

Even though much work has been done to accelerate global illumination for complex environments, very few of them are actually well adapted to architectural environments.

4 Isodata Based Clustering

For constructing clusters, our method makes use of a k-mean type classification algorithm called *isodata* (see [Hall and Ball 1965]). In this section, we first describe this algorithm in 2D space for the sake of clarity then we extend it to 3D space for constructing a set of clusters made up of polygons.

4.1 General 2D case

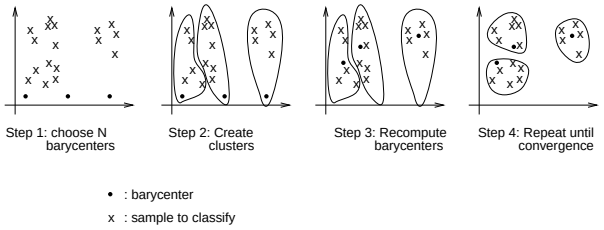

Figure 1: 2D isodata algorithm.

This classification technique uses a set of N barycenters (N being the desired number of clusters) that move among the elements to be classified. Let us work with 2D points on the plane for illustrating the principle. The idea is to associate each group of points with one of these barycenters. During the first iteration, each barycenter is randomly placed on the plane (figure 1, step 1). Then, each 2D point is assigned to its closest barycenter according to a distance function, creating thus an initial set of groups (figure 1, step 2). Then, the actual barycenter is computed for each group and replaces the previous one (figure 1, step 3). The $\mathcal{O}(N_s)^2$, once again, the 2D points are associated with their closest barycenter and so on until convergence. In this case, convergence means the groups of 2D points do not change between two successive steps (i.e. the barycenter has exactly the same computed position). Note that with this algorithm, some barycenters can be associated with no element.

4.2 Convergence issues

It is known that this algorithm may give rise to oscillations with low magnitudes meaning that the convergence criterion is never met. An oscillation corresponds to a situation in which one or more elements to be classified move from one cluster to another continuously. In the case of polygon classification, when this situation takes place, the clusters already meet the classification criterion based on the distance function. This means that as soon as we get an oscillation, we can stop the classification process. From our experiments, when convergence is met, the classification algorithm needs less than 5 iterations. Practically, we stop the classification algorithm after 10 iterations which allow to interrupt a situation of oscillation.

4.3 Extension to 3D polygons

Our aim is to extend this classification technique to 3D polygons. Therefore, we have to define a function giving the distance between a polygon and a barycenter (of a cluster of polygons). This distance function can be any function relying on geometry or photometry (BRDF, colors, textures, etc.).

Our classification algorithm is given in figure 2. We need to specify (i) the notion of barycenter associated with a cluster of polygons, (ii) a function giving the distance between a barycenter and a polygon and (iii) a technique that computes the barycenter of a cluster.

In this algorithm, convergence is met as soon as the groups (clusters in our case), determined at two successive steps, are similar.

Another problem concerns the number of barycenters (or clusters). The user chooses an average number of polygons in a cluster N_{ave}. If N_t is the total number of polygons, the number of barycenters N_g is calculated as : $N_g = N_t / N_{ave}$.

We can use different functions expressing the distance between a polygon and a barycenter (a 3D point). These functions are :

- **Closest vertex :** It chooses the distance between the barycenter and the closest vertex of the polygon.
- **Barycenter :** Determines the distance between the barycenter and the polygon barycenter.
- **Orthogonal :** Computes the orthogonal distance between the barycenter and the polygon plane.
- **Volume increase :** This distance proposed in [Goldsmith and Salmon 1987] favors the smallest bounding box volume increase.

The figure 11 gives an example of the clusters obtained with different average sizes of clusters.

```
Polygon **K-MEANS(N_g, N_p, Polygon polygons[ ]) {
    integer N_g; /* Number of clusters */
    integer N_p; /* Number of polygons to classify */
    Point3D R_k[N_g],R_{k-1}[N_g]; /* Barycenters */
    Polygons C[N_g][N_p]; /* Clusters, N_g = Number of clusters */
    integer J, i, j, k;
    Real D, d[N_p] ; /* d[i] : distance between a polygon i and a barycenter */

    /* Randomly choose a barycenter for each cluster */
    R_k = RandomChoice(N, polygons);
    R_{k-1} = O;

    While R_k ≠ R_{k-1} do {

        /* For each polygon */
        For i = 1 to N_p do {
            D = + INFINITY;

            /* For each barycenter */
            For j = 1 to N_g do {
                d[i] = Distance(R_k[j],polygons[i]);
                /* Memorize the smaller distance and the corresponding cluster */
                if (d[i] < D) then D = d[i] ; J = j;
            }

            /* j is the group index corresponding to */
            /* the barycenter closest to polygons[i]. C[J] is the J^th cluster */
            AddPolygonGroup( polygons[i], C[J] );
        }

        /* The new barycenters are computed */
        R_{k-1} = R_k;
        /* Compute the barycenter for the j^th cluster */
        For j = 1 to N do R_k[j] = barycenter(C[j]);
    }
    return C;
}
```

Figure 2: K-means algorithm. The function *AddPolygonGroup* inserts the polygon i into the group J, while the function *Barycenter* computes the new barycenter associated with the group j.

Among the distance functions we used, the *closest vertex* and the *barycenter* distances provide clusters without any large empty space. The *orthogonal* distance does not provide good results since it is not based on the polygon vertices and the algorithm may group several far-off polygons. The last distance function (*volume increase*) is very sensitive to choice of the initial barycenters. In most cases it provides large clusters with few polygons.

Let us recall that the algorithm works as well with any other distance function. For example, we could create, as in [Funkhouser 1996], clusters of polygons depending on the form factors values.

Another advantage of our method is that the number of clusters may eventually be lower than the number of barycenters (say some of them can be associated with no polygon), which efficiently reduces the number of clusters when several polygons are located in a small subspace.

4.4 Improvements

For each cluster, created with the method described above, we estimate a PVS corresponding to all the clusters potentially visible in the building. Therefore, we construct a bounding box enclosing the cluster. Obviously, when the bounding box is large, so is its PVS. A cluster can have a large size even though it is composed of hundreds of small polygons very close to each other and only one large polygon (see figure 3). This is why we propose to keep in separate clusters the polygons whose smaller edge is larger than a given threshold (1 meter in our program).

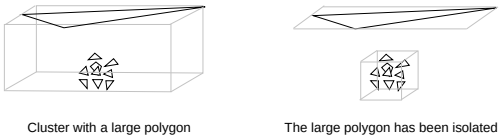

Figure 3: Problem of large polygons.

4.5 Hierarchy construction

One original feature of our clustering method is that it constructs clusters while sparing any cluster hierarchy. However, our classification algorithm can be easily modified for creating a cluster hierarchy as explained hereafter. In this case, a leaf of the cluster hierarchy is a cluster while an internal node is a group of clusters that are close to each other according to the chosen distance function. To create a cluster hierarchy, again we can use the isodata algorithm as explained below. Indeed, a node of a cluster hierarchy may have N children (figure 4, with $N = 2$). N can even be modified at each level.

For complex scenes, the clustering method can be very expensive since the number of elements to be classified is high. The algorithm complexity depends on the number of barycenters and on the number of elements to be classified as well. If N_b is the number of barycenters and N_s the number of elements, there are $N_b \times N_s$ distance computations, with $N_b = N_s/A_g$, where A_g is the average number of polygons per cluster. Consequently, our algorithm complexity is of $\mathcal{O}(N_s)^2$ and compared to the algorithm provided by Goldsmith and Salmon [Goldsmith and Salmon 1987], our method is slower for large sets of objects. However, since clustering is performed for each room in the building the computing time is low. Moreover the construction of a cluster hierarchy easily reduces this complexity down to $\mathcal{O}(N_s.log(N_s))$.

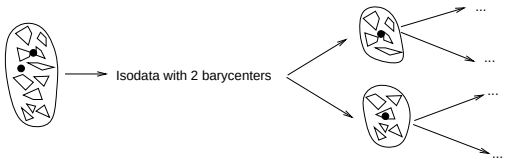

Figure 4: Hierarchy construction with N = 2. For each node, two barycenters (represented by black dots) are chosen.

As an example, we use a binary tree for representing a cluster hierarchy. The first step of the construction of a cluster hierarchy consists in associating all the clusters with the root of the hierarchy. Then, two barycenters are randomly chosen among the clusters and the k-mean algorithm can be directly applied with the clusters as elements to be classified. Every resulting group of clusters is associated with one child of the root and this process is recursively applied to the children nodes in turn.

For storage improvements, the list of clusters and their associated polygons are stored only at the leaves of the hierarchy. This avoids the existence of multiple copies of polygons at the internal nodes.

The figure 14 shows the cluster hierarchy for one room. The objects drawn in red correspond to the part of the hierarchy which is encircled. Let us recall that the leaves hierarchy are clusters.

5 Visibility

In this section, we describe our visibility algorithm. It makes use of constructed clusters and relies on the room portals. Compared to the algorithm proposed by S. Teller in [Teller 1992b], our method is less precise, but very fast and easier to implement.

5.1 Visibility computation

In the same room, we suppose that every cluster can potentially exchange light with every other cluster. Let us consider a cluster C in a room R_a. The problem is to determine the set of clusters visible to C (the PVS) through each portal of R_a (figure 5).

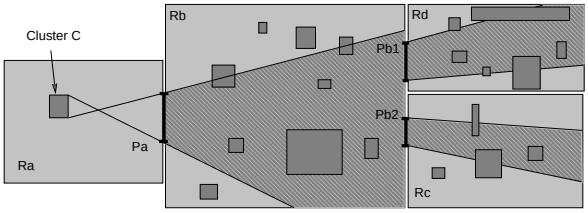

Figure 5: Subspace visible to the cluster C through the portals P_a, P_{b1} and P_{b2}.

With this aim in view, we associate a bounding box with C. For determining the set of clusters visible to it, we compute the subspace potentially visible (S_v) to any 3D point of C through each portal P_i of R_a. When a portal is a rectangle, the supporting plane of which is perpendicular to the ground plane (for us the (x,y) plane of the world coordinate system), this volume is a pyramid made up of 4 half-spaces. Each half-space boundary is a plane that shares one edge with the bounding box of C and one edge with the portal (figure 6). Those edges are coplanar since they are either aligned with one of the 3 axes of the world coordinate system or contained in a plane parallel to (x,y). If the portal is not a rectangle, we consider the portal bounding box. This assumption only enlarges the visibility volume and adds a few clusters to the visibility set of C.

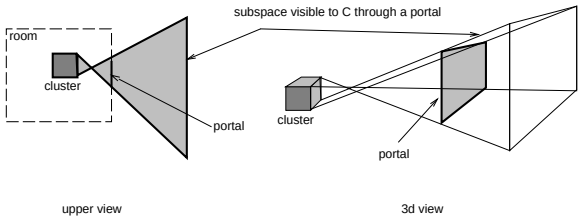

Figure 6: Cluster visibility through one portal. The dark area represents the subspace visible to C through the room portals

The portal P_a is also contained in an adjacent room R_b. The volume S_v contains all the clusters within R_b that are visible to C. Since we are working with convex volumes and half-spaces, finding the clusters inside S_v is a simple task.

Once the clusters, within the adjacent room R_b are determined, it is possible that the visibility volume S_v contains other portals leading to other rooms (for example P_{b1} and P_{b2} in figure 5). In this case, we update the visibility volume through each new portal (for example R_d and R_c) by recursive a application of the process described above (see figure 6). This is in fact a recursive search through all the rooms that are actually visible to the given cluster. The figure 13 shows a cluster (in red) and its PVS (in green on the left and dark blue on the right).

The visibility computation is performed for each cluster of each room. The result (PVS) is a graph where each vertex corresponds to a cluster and each edge to visibility relationships between the corresponding clusters. For this computation, we consider that an edge corresponds to potentially visible clusters. A more precise visibility estimation is made during the radiosity computation for determining patch-to-patch visibility.

6 Radiosity method

As a global illumination system, we use a multi-wavelet radiosity algorithm [Meneveaux et al. 1998a] without any cluster hierarchy nor cluster-based approximations used in [Sillion 1995; Smits et al. 1994]. The initial polygons are recursively divided into surface elements depending on a visibility criterion and an area threshold given by the user. A shooting method is used where each surface element within the selected emitting cluster C_E shoots its light flux to all the visible surface elements belonging to the clusters within the PVS of C_E.

```
global_illumination() {
    typedef struct CLUSTER_FLAGS {
        boolean In_Memory;
        boolean To_Be_Used;
    } ;
    Integer C;        /* Cluster identifier */
    CLUSTER_FLAGS Cluster_In_Memory[number_of_clusters];
    ITERATION = 0;
        /* VG is the visibility graph */
    VG = Read_Visibility_Graph_From_Disk();
    Initialize(Cluster_In_Memory[ ]);
    While not (convergence) AND (ITERATION < Max_Iterations) {
        Unmark_All_The_Clusters();
            /* Choose the first shooting cluster */
        C = Choose_Cluster(VG,ITERATION,Cluster_In_Memory);
            /* Choose_Cluster() modifies Cluster_In_Memory[ ] */
        Mark_Cluster(C);
            /* The cluster C is marked during the current iteration */
        while not (all the clusters are marked) {
                /* The cluster C shoots now its energy */
            Shoot_From_Cluster(C,Cluster_In_Memory);
                /* Choose a shooting cluster which has not already */
                /* shot its energy during this iteration */
            C = Choose_Cluster(VG,ITERATION,Cluster_In_Memory);
            Mark_Cluster(C);
        }
        ITERATION = ITERATION + 1;
    }
}
```

Figure 7: Global radiosity algorithm. The function *Choose_Cluster* selects the cluster according to the used ordering strategy. The function *Initialize* sets the radiosity parameters for all the polygons inside the clusters.

To handle very complex scenes, we have used different ordering strategies described in [Meneveaux et al. 1998a]. The role of an ordering strategy is to select a cluster (or a cell) as emitter, download it together with its PVS, then shoot its energy toward the clusters (or cells) within its PVS. As the whole scene cannot fit in memory, an ordering strategy allows to maintain in memory only a small portion of the scene. The ordering strategies we have used are the following :

- The first strategy (*random*) randomly selects one cluster or cell ;
- The second, called *Max Energy* selects the cluster (or cell) that has the most unshot energy ;
- The *Greedy* algorithms selects the room that entails the minimum of disk access. For the *Greedy S* the disk accesses are given in polygons while for the *Greedy C* this number is given in clusters (or cells).
- *Backtrack* algorithms (we have implemented 2 algorithms called Bactrack *S* and Bactrack *C*) provide disk access estimations of medium range term. Therefore, the algorithms construct a tree (of depth equal to 5) of all the solutions and selects the best path. The disk accesses are given in polygons for the (*backtrack S*) and in clusters (or cells) for the (*backtrack C*).
- The *Traveling Salesman* choice needs a precomputation for estimating the best path passing through all the clusters (or cells) or the nodes of the visibility graph.

7 Results

The results provided in this section have been obtained with a Silicon Graphics R10000 processor, with 380 Mb of memory.

For our tests, the average number of polygons per cluster is 50. Let us recall that a cluster does not contain necessarily 50 polygons (see section 4).

The first test scene is a 3 floors building containing 648 rooms (figure 15 and table 9). Details on our multi wavelet hierarchical radiosity as well as the file formats and datastructure used can be found in [Meneveaux et al. 1998a].

	RND	ME	GS	GC	BS	BC	TS
Cells (rooms)	112h54	132h34	100h46	79h16	108h36	97h12	56h30
Clusters	7h36	9h20	5h21	4h39	45h54	32h30	several days

Figure 8: Computing time with and without clusters for 7 ordering strategies : RND = Random, ME = Max Energy, GS = Greedy S, GC = Greedy C, BS = Backtrack S, BC = Backtrack C, TS = Traveling Salesman

The table 8 gives the computing time for this scene and for 7 ordering strategies. In this table, we can remark that when choosing clusters as emitting groups of polygons rather than cells, the computing time are from far lower, except with the traveling salesman ordering strategy. This can be explained by the fact that the visibility graph contains a high number of nodes (4759 clusters say 4759 nodes). With this strategy, the traversal of this graph, for determining the optimal path visiting all the graph nodes, requires a high computing time.

In what follows, the results have been obtained with the *Greedy C* ordering strategy since it is the most efficient. Moreover, the shooting groups of polygons are clusters.

We have modeled two buildings. The first one, named *Labyrinth* (figure 16), is made up of 182 rooms and 146266 input polygons. The second building is called *Octogon* and is composed of 28 rooms and a total of 277942 polygons. In this last scene, the number of rooms is low (see figure 12) but each room contains a high number

of polygons (some rooms contain more than 60 000 polygons). In this case, as the number of rooms is low and the rooms contain a high number of polygons, the PVS of a cluster contains a high number of polygons. Let us recall that partitioning a scene into cells allows to reduce the time needed for computing clusters. But if the number of polygons in each cell is high, the computing time for clustering still remains high since it is not possible to benefit from the partitioning operation. In this case, the PVS requires a huge amount of memory, which limits our hierarchical radiosity algorithm practically as seen in table 10.

	total # polygons	# of rooms	clustering time	# of clusters	visibility time
Test Scene	58 608	648	15 s	4 759	39 s
Labyrinth	146 266	182	69 s	16 610	60 s
Octogon	277 942	28	120 mn 21 s	3 690	20 s

Figure 9: Used memory and Computing time for the *Greedy C* Strategy.

The table 9 provides, for the 3 test scenes, the computing time for clustering the polygons, the computing time for visibility calculation as well as the number of clusters obtained. For the scene *Octogon*, the number of clusters is low because many polygons are small, close to each other and assigned to a same cluster. This proves the advantage of our clustering method.

	total # polygons	estimated memory	memory w/ rooms	rad/room time	memory w/ clusters	rad/clusters time
Test Scene	58 608	3 Gb	300 Mb	79h16	70 Mb	4h39
Labyrinth	146 266	7.3 Gb	1.5 Gb	–	250 Mb	45h06
Octogon	277 942	13.9 Gb	7.3 Gb	–	5.7 Gb	–

Figure 10: Used memory and Computing time for the *Greedy C* Strategy. Some computations (represented by -) have not been performed because of the large memory needed that was not available on our computer.

With our radiosity algorithm, only a subpart of the scene needs to be stored in memory. Actually, the maximum number of polygons that have to be present in memory during the whole process corresponds to the number of polygons contained in the largest PVS. With the test scene, the largest cell PVS contains 3250 polygons, while the largest cluster PVS contains only 808 polygons. Concerning the *Labyrinth*, the largest cell PVS contains 25000 polygons and the largest cluster PVS contains 3973 polygons. Finally, for the *Octogon* this number of polygons is 146286 for the largest cell PVS and 114542 for the largest cluster PVS. The results in terms of memory given in the table 10 are estimates. Note that the memory size also strongly depends on the number of surface elements.

We can see in the table 9 that for the scene *Octogon* the time needed for computing the clusters gets high compared to those corresponding to the other scenes. In addition, as the scene *Octogon* contains a high number of polygons in each cell (room), the size of the PVS is high in terms of number of polygons and so is the amount of memory needed for storing them (see table 10). For scenes similar to the *Octogon*, cluster-based hierarchical radiosity such as [Sillion 1995; Sauvée 1994] are preferred. Anyway, our methods for clustering and computing visibility are fast and efficient as seen in the table 9.

The table 10 shows that the use of clusters improves radiosity computations drastically, without any loss of precision. However, if the rooms contain a high number of polygons, the PVS still remains high.

8 Conclusion and future works

Solving the radiosity equation for large scenes is time consuming and needs a huge amount of memory. In this paper, we have proposed solutions allowing to reduce the number of polygons stored in memory by using clusters of polygons. Our algorithms have been devised for large buildings, composed of several hundreds of rooms. We have first described a clustering method based on a k-mean type algorithm called *isodata*, taking into account any distance function, based on either geometric or photometric criteria. We have also shown how to construct a cluster hierarchy with only few modifications of the algorithm. Our second contribution concerns a fast and simple technique for determining the set of potentially visible clusters associated with each cluster in the environment. This technique uses the bounding box enclosing the polygons of the cluster as well as the portals (doors or windows) of the buildings. The results show that with our approach it is possible to drastically reduce both the computing time and the amount of used memory.

9 Acknowledgements

Thanks to Aurelien Cirotte who designed most of the furniture used in the Labyrinth and Octogon scenes.

References

AIREY, J. M. 1990. *Increasing Update Rates in the Building Walkthrough System with Automatic Model-Space Subdivision and Potentially Visible Set Calculation*. PhD thesis, University of North Carolina at Chapel hill.

AMANATIDES, J., AND WOO, A. 1987. A fast voxel traversal algorithm for ray tracing. In *Eurographics'87*.

BOUATOUCH, K., AND PATTANAIK, S. N. 1995. Discontinuity meshing and hierarchical multiwavelet radiosity. *Graphics Interface'95*.

CAZALS, F., DRETTAKIS, G., AND PUECH, C. 1995. Filtering, clustering and hierarchy construction: a new solution for ray tracing complex scenes. *Computer Graphics Forum (Eurographics '95) 14*, 3.

CHIN, N., AND FEINER, S. 1989. Near real-time shadow generation using bsp trees. *Computer Graphics 23*, 3 (July), 99–106.

CLEARY, J., AND WYVILL, G. 1988. Analysis of an algorithm for fast distributed ray tracing using uniform space subdivision. *The Visual Computer 4*, 2 (July), 65–83.

COORG, S., AND TELLER, S. 1997. Real-time occlusion culling for models with large occluders. *ACM Symposium on Interactive 3D graphics.*.

DRETTAKIS, G., AND FIUME, E. 1994. A fast shadow algorithm for area light sources using backprojection. *ACM SIGGRAPH'94*.

DRETTAKIS, F. D. G., AND PUECH, C. 1997. The visibility skeleton: A powerful and efficient multi-purpose global visibility tool. *SIGGRAPH'97*.

FUNKHOUSER, T. 1996. Coarse-grained parallelism for hierarchical radiosity using group iterative methods. *ACM SIGGRAPH'96 proceedings* (Aug.), 343–352.

GIBSON, S., AND HUBBOLD, R. 1996. Efficient hierarchical refinement and clustering for radiosity in complex environments. *Computer Graphics Forum 15*, 297–310.

GIGANTE, M. 1990. Accelerated ray tracing using non-uniform grids. In *AusGraph'90*.

GOLDSMITH, J., AND SALMON, J. 1987. Automatic creation of object hierarchies for ray tracing. In *IEEE Computer Graphics and Applications*, vol. 7, 14–20.

GORAL, C., TORRANCE, K., GREENBERG, D., AND BATTAILE, B. 1984. Modeling the interaction of light between diffuse surfaces. *Computer Graphics 18*, 3 (July), 213–222.

GORTLER, S. J., SCHRODER, P., COHEN, M. F., AND HANRAHAN, P. 1993. Wavelet radiosity. In *Computer Graphics Proceedings, Annual Conference Series*, ACM, Ed., 221–230.

HABER, J., STAMMINGER, M., AND SEIDEL, H.-P. 2000. Enhanced automatic creation of multi-purpose object hierarchies. In *Pacific Graphics'2000*, 52–61.

HAINES, E. A., AND WALLACE, J. R. 1991. Shaft culling for efficient ray-cast radiosity. In *Proceedings of 2nd Workshop on Rendering*, 122–138.

HALL, D. J., AND BALL, G. H. 1965. Isodata a novel method of data analysis and pattern classification. Tech. Rep. 5 RI project 5533, Stanford Research Institute, CA, USA.

HANRAHAN, P., SALZMAN, D., AND AUPPERLE, L. 1991. A rapid hierarchical radiosity algorithm for unoccluded environments. *Computer Graphics 25*, 4 (July), 197–205.

HASENFRATZ, J.-M., DAMEZ, C., SILLION, F., AND DRETTAKIS, G. 1999. A practical analysis of clustering strategies for hierarchical radiosity. In *Computer Graphics Forum (Proc. of Eurographics '99)*, P. Brunet and R. Scopigno, no. 18(3), 221–232.

JOHN M. AIREY, JOHN H. ROHLF, F. P. B. 1990. Towards image realism with interactive update rates in complex virtual building environments. *ACM Siggraph* (May), 41–50.

KAY, T. L., AND KAJIYA, J. T. 1986. Ray tracing complex scenes. *ACM SIGGRAPH'86.*, 269–278.

KLIMASZEWSKI, K. S. 1997. Faster ray tracing using adaptive grids. *IEEE Computer Graphics and Applications 17*, 1 (Jan.), 42–51.

LEBLANC, L., AND POULIN, P. 2000. Guaranteed occlusion and visibility in cluster hierarchical radiosity. In *Eurographics Workshop on Rendering 2000*, 89–100.

MENEVEAUX, D., AND BOUATOUCH, K. 1999. Synchronisation and load balancing for parallel hierarchical radiosity of complex scenes on a heterogeneous computer network. *Computer Graphics Forum 18*, 4 (Dec.).

MENEVEAUX, D., BOUATOUCH, K., AND MAISEL, E. 1998. Memory management schemes for radiosity computation in complex enviroments. In *Computer Graphics International*.

MENEVEAUX, D., MAISEL, E., DELMONT, R., AND BOUATOUCH, K. 1998. A new partitioning method for architectural environments. *Journal of Vizualisation and Computer Animation*, 3148 (Nov.).

MÜLLER, G., SCHFER, S., AND FELLNER, D. W. 2000. Automatic creation of object hierarchies of radiosity clustering. *Computer Graphics Forum 19*, 4 (Dec), 213–221. Sabine Coquillart, David J. Duke: Editorial. 194.

SALAM, I., NEHLIG, P., AND ANDRES, E. 1999. Discrete raycasting. In *8th International Workshop on Discrete geometry for Computer Imagery (DGCI'99)*.

SAUVÉE, M. 1994. Maillage de discontinuités pour le modèle de radiosité. Tech. rep., IRISA, Sept.

SILLION, F., AND DRETTAKIS, G. 1995. Feature-based control of visibility error: A multi-resolution clustering algorithm for global illumination. *ACM SIGGRAPH'95*.

SILLION, F. 1994. Clustering and volume scattering for hierarchical radiosity calculations. *5th Eurographics Workshop on Rendering*.

SILLION, F. 1995. A unified hierarchical algorithm for global illumination with scattering volumes and objects clusters. *IEEE Transaction On Graphics 1*, 3.

SMITS, B., ARVO, J., AND GREENBERG, D. 1994. A clustering algorithm for radiosity in complex environments. In *Computer Graphics Proceedings, Annual Conference Series*, 435–442.

SNYDER, J., AND BARR, A. 1987. Ray tracing complex models containing surface tesselations. In *Computer Graphics ACM SIGGRAPH*, 119–128.

T. FUNKHOUSER, S. TELLER, C. S., AND KHORRAMABADI, D. 1996. The uc berkeley system for interactive visualization of large architectural models. *Presence 5*, 1, 13–44.

TELLER, S., AND HANRAHAN, P. 1993. Global visibility algorithms for illumination computations. In *Computer Graphics Proceedings, Annual Conference Series*, 239–246.

TELLER, S., FOWLER, C., FUNKHOUSER, T., AND HANRAHAN, P. 1994. Partitioning and ordering large radiosity computations. In *Computer Graphics Proceedings, Annual Conference Series*, 443–450.

TELLER, S. 1992. Computing the antipenumbra of an area light source. In *SIGGRAPH'92*.

TELLER, S. J. 1992. *Visibility Computations in Density Occluded Polyhedral Environments*. PhD thesis, University of California at Berkeley.

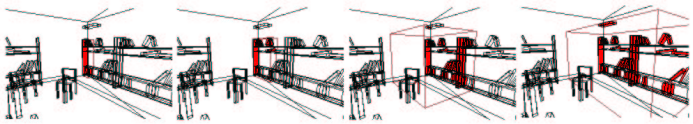

Figure 11: 5, 10, 50, 150 polygons per group, with the barycenter distance. The cluster bounding box is drawn in red as well as the polygons it contains.

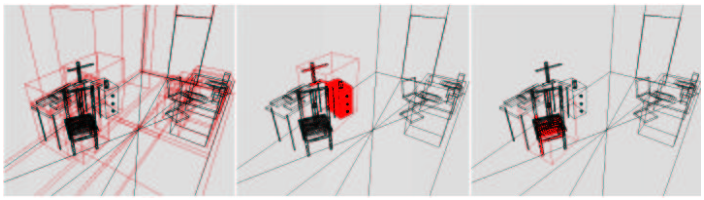

Figure 12: Clusters with a high number of polygons.

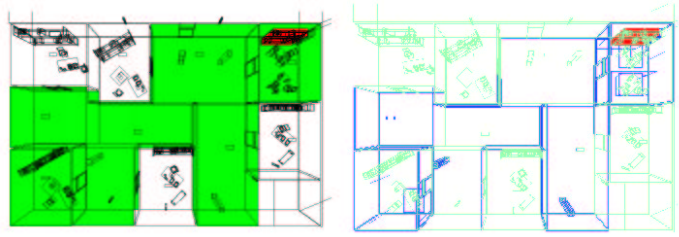

Figure 13: Clusters visible (in green on the left and blue on the right) from one cluster (in red) in the whole building. Note that the whole rooms are not visible, but the floors, ceilings and walls are also taken into account.

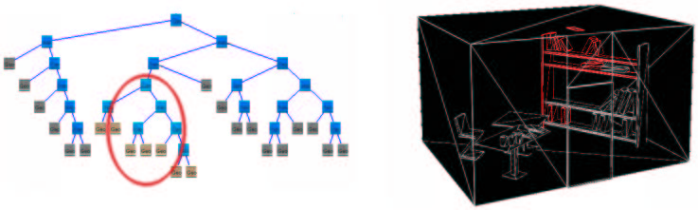

Figure 14: Resulting hierarchy for one room.

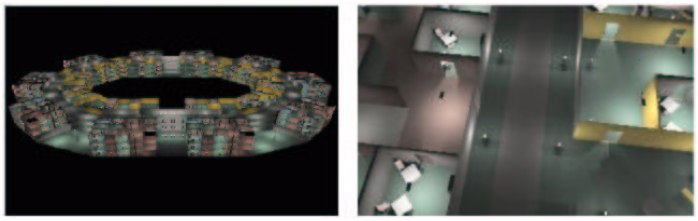

Figure 15: Images of the test scene.

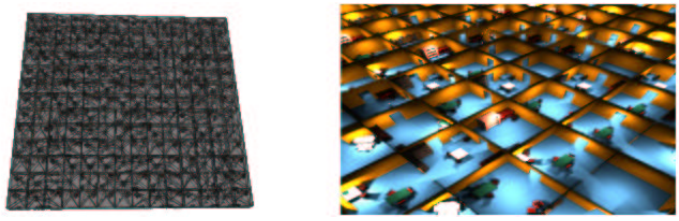

Figure 16: Images of the Labyrinth.

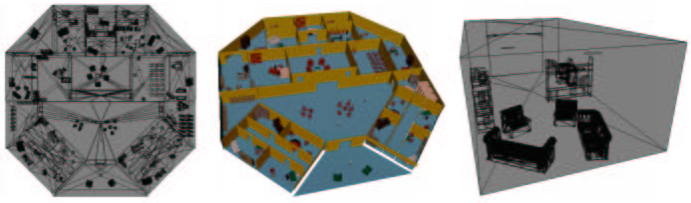

Figure 17: Images of the Octogon.