Efficient Partitioning of architectural Environments

Daniel Meneveaux, Eric Maisel and Kadi Bouatouch
IRISA, Campus de Beaulieu, 35042 Rennes Cedex, France

Abstract

Computing global illumination in a moderate time for complex environments and walking through them is one of the challenges in computer graphics. To meet this goal, a preprocessing is necessary. This preprocessing consists in partitioning the environment into cells and determining visibility between these cells. Most of the existing partitioning methods relie on the Binary Space Partitioning technique (BSP) and are often applied to axial environments. This technique entails a too important number of cells which do not necessarily fit with the topology of the environment. We propose in this paper a model-based partitioning method which can be applied to non axial buildings. It results in a few number of cells fitting at best with the environment topology.

1 Introduction

Even though the throughput of the graphics hardware has improved drastically over the past decade, for complex environments such as buildings containing millions of polygons, real time rendering (for walkthrough) still remains impossible [1, 2] and global illumination (with the radiosity method) still is a very demanding process in terms of computation and memory resources [3]. To overcome this problem a preprocessing for limiting the number of polygons is needed. For example, when walking through an environment only the polygons (or geometric primitives) visible from the viewpoint are rendered at each frame. As for global illumination with the radiosity method, only a subset of polygons contributes to the illumination of a polygon within the scene.

This preprocessing relies on the subdivision of the scene into cells. A viewpoint or a polygon in a cell C sees only the polygons lying in C as well as those visible through the holes (also termed *portals* by several authors) in its boundary, like windows and doors for bulding interiors.

The subdivision process is often based on a binary space partitioning (BSP). Indeed, the scene is recursively split with the help of planes containing a polygon. The result is a set of 3D cells. The number of cells may be too important and in general do not fit with the topology of the scene. For example, a room may be subdivided into several pieces. Furthermore, the BSP method has been applied only to axial complex environments (all the scene polygons are normal to one of the coordinate system axis) for a reason of simplicity [4, 5]. Indeed, the partitioninig with a BSP technique is very difficult for non axial scenes.

The main contribution of our paper is a novel partitioning method which can be applied to non axial buildings. Compared to the BSP technique, our subdivision method results in a fewer number of cells that better fit with the scene topology.

In this paper previous related works are first presented followed by an overview of our partitioning method and by details on its implementation. Next, results and comparison with Airey's method are given to demonstrate the efficiency of our partitioning method.

2 Previous Works

The most important works dealing with visibility calculation in complex environments (buildings) are due to Airey [4] and Teller [5]. Both works relie on a BSP subdivision of the environment into 3D cells. For each polygon P (or cluster O, i.e. object made up of polygons) within a cell C, the polygons visible to P (or to O) are determined and forms the PVS (potentially visible set). The subdivision into cells together with the resulting PVS's allow to walk through a complex building and to make possible the computation of radiosity solutions.

When a BSP scheme is used to partition the scene, the choice of the best splitting plane is primordial. Note that a splitting plane contains at least one polygon. Airey devised a heuristic function to choose the splitting planes. This function combines three criteria quantified by:

- the *balance* factor of the split: it shows how evenly the plane separates the scene;
- the *occlusion* factor: it provide an information on how well the plane hides the two sides from each other;
- the *split* factor: it shows how little the plane splits the scene polygons.

The heuristic function proposed by Airey is a linear combination of these three factors:

$$partition\_function = 0.5 * occlusion + 0.3 * balance + 0.2 * split$$

.

To make the BSP partitioning method tractable, Teller and Airey used only axial buildings, which limits their method practically. Another drawback is that the BSP technique results in a high number of cells which may not fit with the topology of the scenes. Indeed, a room or a corridor may be split into several pieces. The consequence of this is the large required memory storage and the inefficiency of the visibility calculation.

For these reasons, we propose in this paper a new partitioning method that may handle non-axial complex buildings and fits at best with the topology of the scene. With this method walkthrough and radiosity calculation become more efficient.

3 Overview

Only building interiors are considered as scenes. These latter are modeled with planar convex polygons. No a priori information is known about the scene: the user has not to distinguish the walls from the furnitures. The scene is not necessarily axial.

To partition the scene into 3D cells, our method utilizes only the vertical polygons, say the ones normal to the floor. In a preprocessing step, the set of polygons is divided into two lists: one containing the vertical polygons and the other the rest of the polygons. Our partitioning method operates in a 2D space corresponding to the horizontal plane of the WCS (world coordinate system).

By convention, the horizontal plane contains the x and y axes and will be denoted Oxy from now on. With this aim in view, the vertical polygons are projected onto the Oxy plane. This projection results in a set of segments which are represented by points in the dual space (θ, ρ). In this dual space, a segment is represented by a point (θ, ρ), where ρ is the orthogonal distance of the WCS origin to this segment, and θ the angle formed by this segment and the x axis. Neighbouring points are clustered and each cluster is represented by one point R. The vertical polygon corresponding to R is chosen as splitting plane. Once the splitting planes have been determined, the cells are determined with the help of construction rules, for example a bedroom is rectangular and its width has a value ranging from 4 meters to 7 meters. These rules guide the construction of the cells by means of the spliting planes. The next step of our method is the determination of visibility relationships between the resulting cells. To this end an adajacency graph and a visibility graph are determined. A cell views another cell through portals like windows and doors. We have devised a simple and efficient method to determine these portals. The result of the proposed partitioning method is a reduced number of cells which fit at best with the topology of the scene.

The next sections give more details on the determination of the splitting planes, the associated cells and of the adjacency and visibilty graphs. The following sections deal with the implementation of our method and show some results.

4 Determining the splitting planes

4.1 Dual space

As said before, only the vertical polygons contribute to the scene partitioning into 3D cells. Each vertical polygon is projected onto the Oxy horizontal plane. The resulting segment belongs to the line defined by the equation :

$$N \bullet P + \rho = 0,$$

where $P(x, y)$ is a point lying on the line, N the normal to the polygon and ρ the orthogonal distance of the WCS origin to the line (or to the polygon). Each segment is represented by a point in the dual space (θ, ρ), where $\theta = \arccos(N_x) * sign(N_y)$, N_x and N_y being the x and y coordinates of the normal to the polygon (see figure 1).

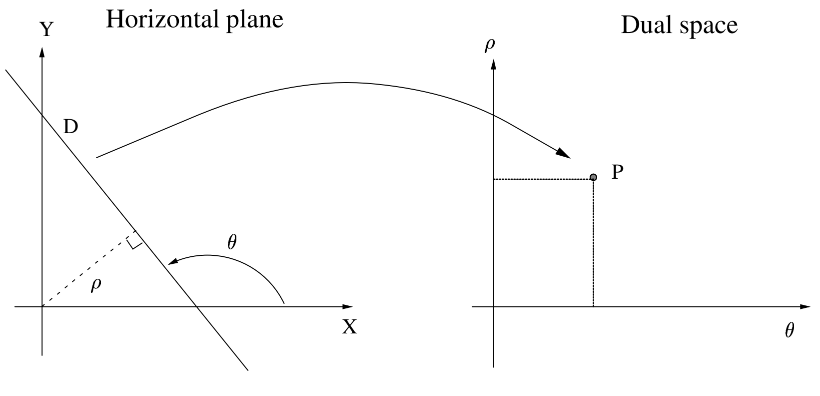

Figure 1: Dual space

The data structure associated with a vertical polygon is by figure 2.

```
typedef struct POLYGON {
    Color *color;                    /* color */
    int nb_vertices;                 /* number of vertices */
    Point points[MAXPOINTS];         /* List of vertices */
    Vector normal;                   /* normal */
    double area;                     /* area of the polygon */
    double theta;                    /* theta angle */
    double rho;                      /* rho */
    double abscissa[2];              /* Coordinates in the 1D LCS of the */
                                     /* projected vertical polygon */
} *Polygon;
```

Figure 2: Data structure for a vertical polygon

This data structure contains the coordinates (θ, ρ) in the dual space of the segment resulting from the projection of the polygon onto the Oxy horizontal plane.

A 1D local coordinate system (called LCS from now on) is associated with the line supporting this segment and has as origin the orthogonal projection of the WCS origin onto this line as shown in figure 3. The direction vector of this coordinate system is defined by the angle $\theta = \theta_N - \pi/2$, where θ_N is the angle formed by the normal to the polygon and the x axis. The coordinate of each segment endpoint is given by the field *abscissa* as pointed out in the above data structure.

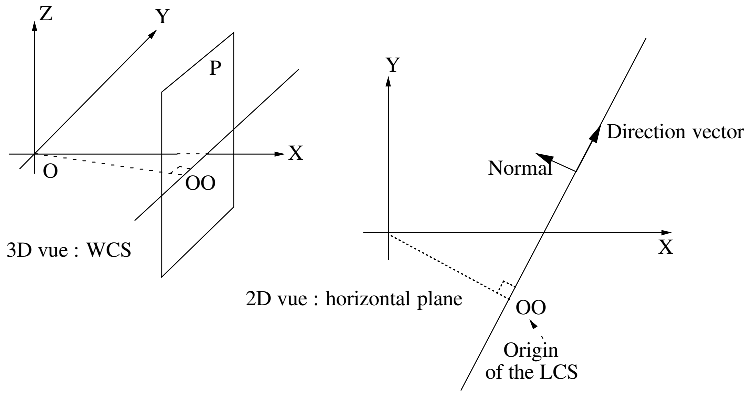

Figure 3: 1D local coordinate system (LCS)

4.2 Clustering

Once all the vertical polygons have been projected onto the Oxy horizontal plane, their projections (segments) are transformed into points in the dual space (θ, ρ). The aim now is to cluster neighboring points. The result is a certain number of clusters. From each cluster we determine a point (θ_s, ρ_s) which corresponds to the projection of a vertical splitting plane P_s.

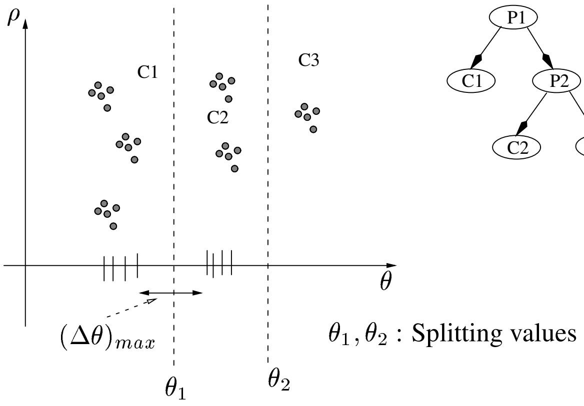

Figure 4: Classification according to θ

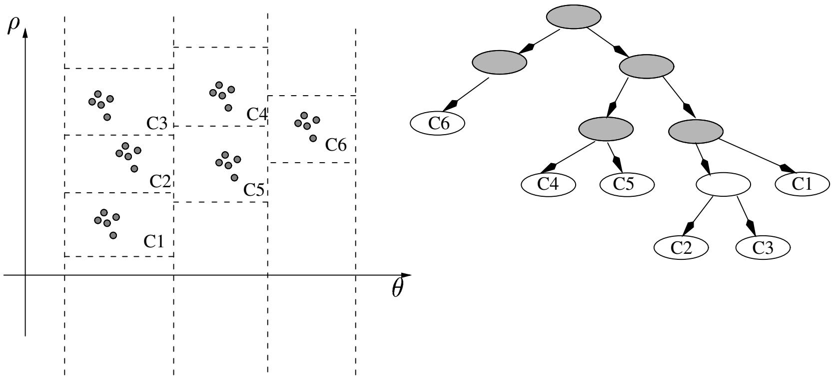

Figure 5: Classification according to ρ

Let us see now how the clustering is performed. Recall that a vertical plane as well as the associated dual coordinates (θ, ρ) are described by the same data structure *Polygon* given above. First of all, the vertical polygons are sorted according to their θ value. The obtained sorted list is recursively split into two parts, which results in a binary tree. The splitting value θ_s is equal to $\theta_i + 0.5 * max(\theta_{i+1} - \theta_i)$, where θ_i and θ_{i+1} are the θ values of two subsequent vertical polygons in the sorted list. The recursion stops when $max(\theta_{i+1} - \theta_i)$ is below a certain threshold (figure 4). A leaf of this tree is a list of vertical polygons whose $\theta' s$ are close to each other. Each leaf is sorted according to the ρ value and recursively split into two parts. The splitting value ρ_s is equal to $\rho_i + 0.5 * max(\rho_{i+1} - \rho_i)$. The recursion stops when $max(\rho_{i+1} - \rho_i)$ is below a certain threshold (figure 5). The result is given in figure 5. Each leaf is a cluster of vertical polygons with very close ρ and θ values.

4.3 Splitting planes

For each cluster we determine a splitting plane whose equation is $N_s \bullet P + \rho_s = 0$. The associated θ_s value must be chosen as close as possible to those of the polygons in the cluster while favouring the polygon of greatest area. To achieve this, we use :

$$\theta_s = \frac{\sum_{i \in cluster} area_i * \theta_i}{\sum_{i \in cluster} area_i}.$$

Furthermore, this plane must be chosen so that the half-space D defined by $N_s \bullet P + \rho_s > 0$ contains all the polygons belonging to the cluster. This condition can be expressed as $\rho_s > max(-N_s \bullet P_i)$, where the $P_i^\prime s$ are the vertices of the polygons within the cluster. Figure 6 gives the data structure defining a splitting plane.

```
typedef struct SPLITTING_PLANE {
    double theta_s;
    double rho_s;          /* Distance a l'origine */
    double area;           /* Sum of the areas of the polygons in
                              the associated cluster*/
    vector Normal_s        double origin[2];        /* Origin of the LCS */
    double abscissa[2];       /* minimum and maximum coordinatee
                              of the polygons in the LCS*/
} *Splitting_plane;
```

Figure 6: Data structure for a splitting plane

LCS is the 1D local coordinate system associated with the splitting plane. The segments resulting from the projection of the vertical planes (within the same cluster represented by the splitting plane) onto the horizontal plane Oxy are projected in turn onto this LCS axis. Abscissa are the coordinates (called Min and Max) of the endpoints of the smallest segment (lying on this axis) containing all these projected segments (see figure 7). From now on, this segment will be called MinMax segment. So, a MinMax segment is associated with each splitting plane.

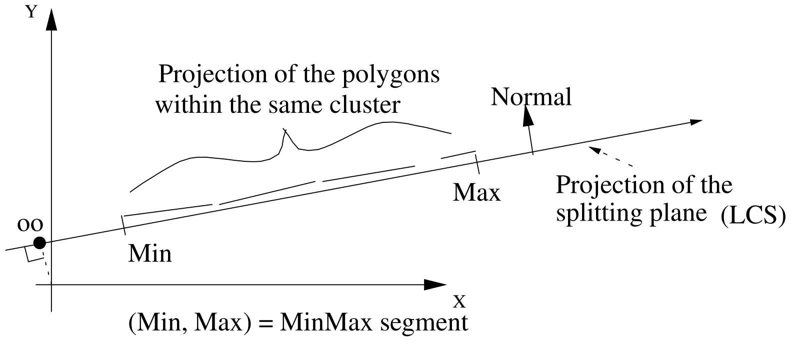

Figure 7: MinMax Segment

5 Cell construction

Once all the splitting planes have been determined, the partitioning of the scene into cells is performed. As with each splitting plane is associated a half-space defined by $N_s \bullet P + \rho_s > 0$, a cell is no more than the intersection of such half-spaces. The normal to each splitting plane points toward the cell interior. Since our objective is to partition the scene into a reduced number of cells fitting at best the scene topology, the partitioning process is guided by some a priori architectural knowledges. For example, most of the rooms are rectangular and have a width larger than one meter and smaller than ten meters. Some rooms are delimited by two parallel walls...

Before showing how these rules are used, let us give some definitions. A splitting plane $P1$, defined by (θ_1, ρ_1) in the dual space, faces another splitting plane $P2$, defined by (θ_2, ρ_2), if $\theta_1 = \theta_2 + \pi$ and $\rho_1 + \rho_2 > 0$, and if their MinMax segments overlap (see figure 8). Two MinMax segments overlap if the orthogonal projection of one projects partially or completely on the other.

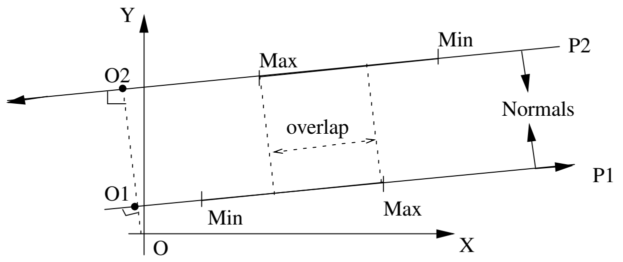

Figure 8: Two splitting planes facing each other

5.1 Applying the construction rules

The used rules help determining several kinds of cells.

5.1.1 Rectangular cell

In a building, most of the rooms, offices and corridors are rectangular and have a width and a length whose values range over $[Size_{min}, Size_{max}]$. By exploiting this remark we can extract 3D rectangular cells corresponding to rooms with 4 walls as explained in figure 9.

1- Choose the most occlusive splitting plane P_0, i.e. whose area field defined in its associated data structure is maximum
2- Let P_1 be a splitting plane facing P_0 so that $Size_{min} \leq \rho_1 + \rho_2 \leq Size_{max}$
3- Choose a splitting plane P_2 perpendicular to P_0, i.e $\theta_2 = \theta_0 + \pi/2$
4- Choose P_3 facing P_2 so that $Size_{min} \leq \rho_2 + \rho_3 \leq Size_{max}$

Figure 9: Rectangular cell with 4 walls

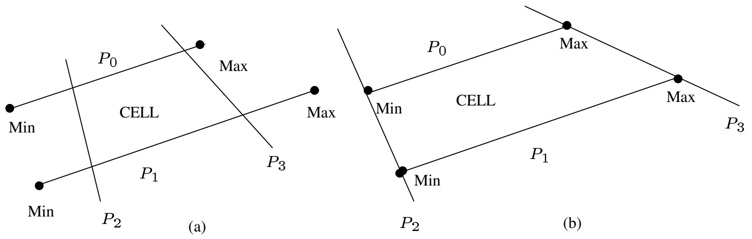

Figure 10: Cell with two parallel walls

5.1.2 Cell with 2 parallel walls

Now the objective is to extract cells defined by at least two parallel splitting planes (2 parallel walls) as shown in figure 10. Figure 11 gives the algorithm allowing the extraction of these kinds of cell.

1- Choose 2 parallel splitting planes P_0 and P_1 facing each other whose MninMax segments are respectively M_0 and M_1 2- Find 2 other splitting planes P_2 and P_3 whose projections onto the horizontal plane intersect M_0 and M_1 3- If these planes exist then construct the cell (figure 10-a) 4- Else create them as planes passing through the Min and Max endpoints of M_0 and M_1 (figure 10-b)

Figure 11: Cell with 2 parallel walls

5.1.3 Other convex cells

Once all the rectangular cells as well as those defined by at least two parallel splitting planes have been determined, we suppose that the remaining cells are convex, i.e. their projections onto the horizontal plane are triangles or convex polygons. Let us consider the example given in figure 12 to explain how these convex cells are extracted. This figure shows the orthogonal projection of the splitting planes onto the horizontal plane. First of all we choose the most occlusive splitting plane P_0 whose Minmax segment has Min and Max as endpoints. Then we look for the closest splitting plane P_1 to the point Min. Next, we repeat the same process by searching for the plane $P2$ which is closest to the point Min of the $P1$ MinMax segment. In the same manner we find the closest plane P_3 to the point Min of the $P2$ MinMax segment. As P_3 intersects the initial splitting plane P_0 the cell searching process ends up and the resulting cell (shaded in the figure) is delimited by the planes : P_0, P_1, P_2, P_3.

5.2 Guiding the search for cells

For a reason of efficiency it is worth to guide the cell search process. With this aim in view, we propose a method wich has been applied only for rectangular cells. This method operates as follows

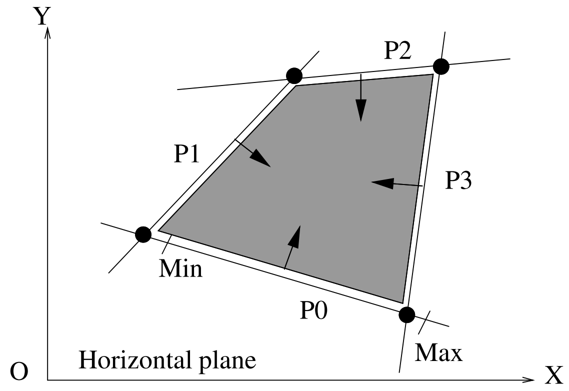

Figure 12: Other convex cells

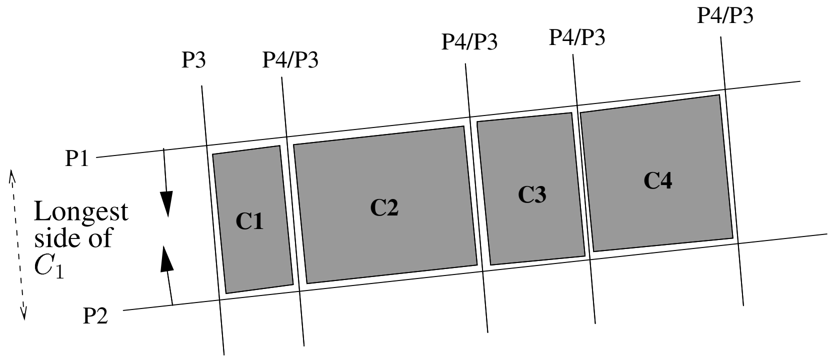

Figure 13: Guided cell search

(see figure 13). Suppose a rectangular cell C_1 has been determined, and let P_1, P_2, P_3, P_4 be the associated splitting planes. First, we choose one of these planes lying along the longest side of C_1, for example P_4. To extract a new cell C_2 we keep P_1 and P_2 and consider $P4$ as the new third plane. $P4$ becomes P_3 for C_2. Now, we only have to determine the new plane P_4 facing P_3. This process is repeated till all the cells are extrated : C_3, C_4...

6 Adjacency and Visibility graphs

In the adjacency graph a vertex is a cell and an edge between two vertices means that there exists at least one portal between the adjacent cells corresponding to these vertices. Let us see now how these portals are determined.

6.1 Determining the portals

A cell views another cell through portals such as windows and doors. The problem of determining such portals has already been addressed by Airey [4]. Airey extends and generalizes the plane-sweep triangulation algorithm to triangulate a region of the plane defined by set operations such as: union, intersection and difference. As pointed out by Airey, this approach is quite complicated and time-consuming.

For this reason, we have devised a simple and efficient method for determining the portals as explained hereafter. Our method is capable of computing even non-convex portals.

Recall that a cell C is delimited by splitting planes whose normal points toward the interior of C. Let P_s be a splitting plane. It then partitions the space into two half-spaces: V_+ and V_-. Let V_+ be the hal-space containing the cell. If a scene polygon P is such that some of its vertices are in V_+ and the others in V_- then it is clipped by P_s, which results in a segment called *portal segment* from now on. A portal is delimited by a subset of these segments. Furthermore, the vertices of each object polygon are ordered so that its normal points toward the outside of the object containing it. A polygon P clipped by a splitting plane P_s is divided into two parts: P_{in} (inside the cell) and P_{out} (outside the cell). let S be the resulting portal segment. If we visit the vertices of P_{in} together with those of S in the sense defined by the orientation of P, S becomes automatically oriented (see figure 15 and 16). In this way all the portal segments become oriented, which facilitates the construction of the portals as shown in figure 14. Note that this portal construction procedure is applied to two neighbouring cells sharing the same splitting plane but with opposite orientation.

```
typedef struct VECTOR {
    point : A        /* starting point */
    point : B        /* end point */
} portal_segment        /* vector AB */
C : List_of_portal_segments = empty;        /* delimiting a portal */;
L : List_of_portal_segments = empty;
S_0, S_1, S_first : portal_segment;
while L ≠ empty do
{
    Take a portal segment S_0;
    S_first = S_0;
    L = L − {S_0};
    C = C + {S_0};
    Portal_found = false;
    while not Portal_found do {
        Find the segment S_1 so that S_1.A = S_0.B;
        L = L − {S_1};
        C = C + {S_1};
        S_0 = S_1;
        if S_0.B = S_first.A then Portal_found = true;
    }
}
```

Figure 14: Determining the portals

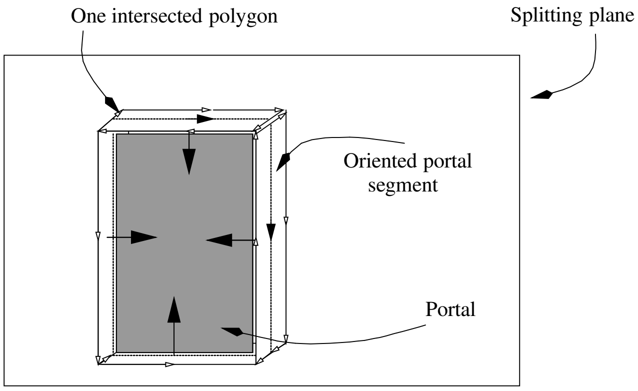

Figure 15: Portal in 3D

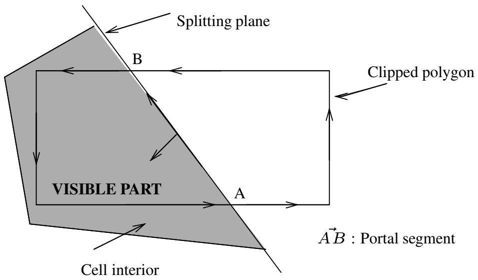

Figure 16: Portal segments

6.2 Visibilty graph

First of all, the adjacency (or connectivity) graph G_a is constructed. In this graph, a vertex is a cell while the existence of an arc between two vertices means that the associated cells are adjacent and visible from each other through at least one common portal. As for the visibility graph G_v, it provides information on the visibility between two cells non necessarly adjacent. A vertex of G_v corresponds to a cell. An edge (of G_v) links two cells C_i and C_j if there exists at least one point I lying on a portal of C_i and a point J on a portal of C_j such that I views J through a sequence of portals $O_1, O_2, ...O_n$ located between C_i and C_j. The portal sequence $O_1, O_2, ...O_n$ is determined with the help of a recursive traversal of the adjacency graph.

6.3 Discussion

Presently, our partitioning method is capable of handling only three kinds of construction rules allowing the extraction of rectangular cells, cells with two parallel walls and convex cells. Rules for extracting other kinds of cells are under investigation. Furthermore, once a cell has been extracted, all the scene polygons within this cell (such as furnitures) are determined. This is simply done by checking if such polygons are in the half-spaces pointed by the normals to the splitting polygons delimiting the cell. Note that some polygons may be subdivided into pieces. In this way, the ceilings (horizontal polygons) and the roofs (oblique polygons) are accounted for even though they are not vertical. They are just added to the polygons within the cell. The result of this is a certain number of cells of different kinds as well as their contents.

7 Results

This section shows some results obtained with our partitioning method and with the one proposed by Airey and Teller which is based on a binary space partitioning. Two scenes have been considered. The first one represents the first floor of the *Soda hall* modeled by Teller and the second, named *Irisa*, corresponds to the floor occupied by our research group. Note that *Irisa* is a non-axial scene.

Soda hall is a scene composed of 2003 rectangular polygons and does not contain any furnitures. It is made up of 150 rooms and corridors. All the the most occlusive polygons (walls) are axial, say perpendicular to one of the coordinate system axes. This scene has been partitioned with our method and with the BSP technique proposed by Airey in which the splitting planes are determined by a heuristic function combining three criteria (see section 2) and the portals (rectangular in this case) has been determined with set operations (difference only).

Table 1 gives some results on the *Soda hall* scene obtained with both methods.

	BSP	Our method
Partitioning time	5 s	141 s
Number of cells	1528	155
Portal computing time	12 s	< 1s
Visibility graph computing time	310 s	< 1s

Table 1: Comparision with the BSP technique

This table shows that the number of cells obtained with the BSP technique is equal to ten times the one obtained with our method. Indeed, some cells created by the BSP technique, like those located between two adjacent rooms and due to the thickness of the walls, are useless. This kind of cell cannot be extracted by our partitioning method. The partitioning time is more important with our method but the time for computing the portals is far lower. Figure 17 represents a top-view of the partitioning result with our method, a unique color is associated with each cell.

Irisa is a furnished scene composed of 6774 polygons and contains 50 rooms. There are two portals per cell on average. Table 2 gives some results obtained with our method. A top-view of the resulting cells is given by figure 19, a unique color is associated with each cell and its furnitures. In figure 20 the furnitures are displayed in a wire-frame form (the triangulation we can observe in the image is due to the *Performer* tool). We can remark that the resulting cells fit with the topology of the scene wich explains the low number of cells. Like for scene *Soda hall*, the time for computing the portals is insignificant.

Recall that with each vertical scene polygon is associated a point in the 2D dual space. This point is assigned the area of this polygon. Once the clustering has been performed, each cluster is in its turn assigned an area value equal to the sum of areas associated with its points. These area values are shown in figure 18 for scene *Irisa* and correspond to height values. The more a group of polygons is occlusive the larger is the area value of its associated cluster. In this figure there broadly exist two kinds of cluster: clusters with a low height and clusters with a high height. The low height clusters are randomly distributed. They correspond to the least occlusive groups of polygons such as furnitures. The other clusters are regularly distributed, they represent the walls. Note that the existence of two kinds of clusters makes easy the search for splitting planes.

Partitioning time	2 mn 08 s
Number of cells	51
Portal computing time	< 1s
Visibility graph computing time	< 1s

Table 2: Results with the *Irisa* scene

8 Conclusion

A new algorithm for partitioning architectural environments (building interiors) has been described in this paper. Unlike the methods proposed in the literature, the algorithm does not make use of a binary space partitioning scheme but a dual space which represents the most occlusive polygonal objects. The cells are determined according to construction rules defined by cell models. This model-based approach allows to avoid the creation of cells which do not fit with the topology of the scene. In addition, it is easier and more intuitive to relie on geometric rules rather than tuning parameters (such as the ones described in section 2) empirically. Furthermore, the addition of new construction rules, and consequently new cell models entails a very slight modification of the algorithm. Another advantage of the algorithm is its capability of similarly handling axial and non-axial scenes. As for the determination of the portals, this algorithm makes use of a simple and efficient method. We are already using this algorithm for walkthrough and radiosity calculation in complex architectural environments.

One of the perspectives of this work is the extension of the proposed partitioning method to outdoor scenes like urban environments and to digital elevation maps (DEM). For example, in the case of urban environments the streets and squares are cells and the façades are the cell boundaries. Another direction to be invistigated consists in finding out a dual space so as to handle oblique occlusive polygons.

References

[1] Frederick P. Brooks John M. Airey, John H. Rohlf. Towards image realism with interactive update rates in complex virtual building environments. *ACM Siggraph*, pages 41–50, May 1990.

[2] Thomas Funkhouser, Seth Teller, and Carlo Sequin. The uc berkeley system for interactive visualization of large architectural models. *Presence*, 5(1):13–44, 1996.

[3] Seth Teller & Celeste Fowler & Thomas Funkhouser & Pat Hanrahan. Partitioning and ordering large radiosity computations. In *Computer Graphics Proceedings, Annual Conference Series*, pages 443–450, 1994.

[4] John M. Airey. *Increasing Update Rates in the Building Walkthrough System with Automatic Model-Space Subdivision And Potentially Visible Set Calculations*. PhD thesis, University of North Carolina at Chapel Hill, 1990.

[5] Seth Jared Teller. *Visibility Computations in Density Occluded Polyhedral Environments*. PhD thesis, University of California at Berkeley, 1992.

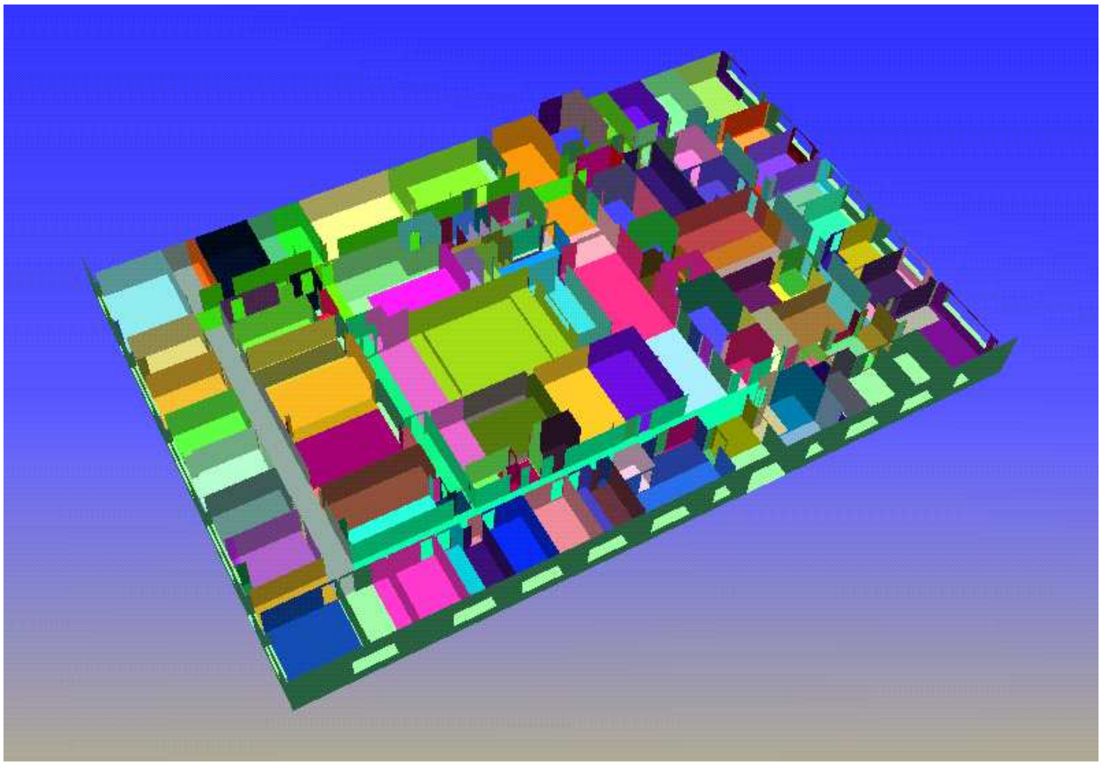

Figure 17: top-view of *Soda hall* with our method

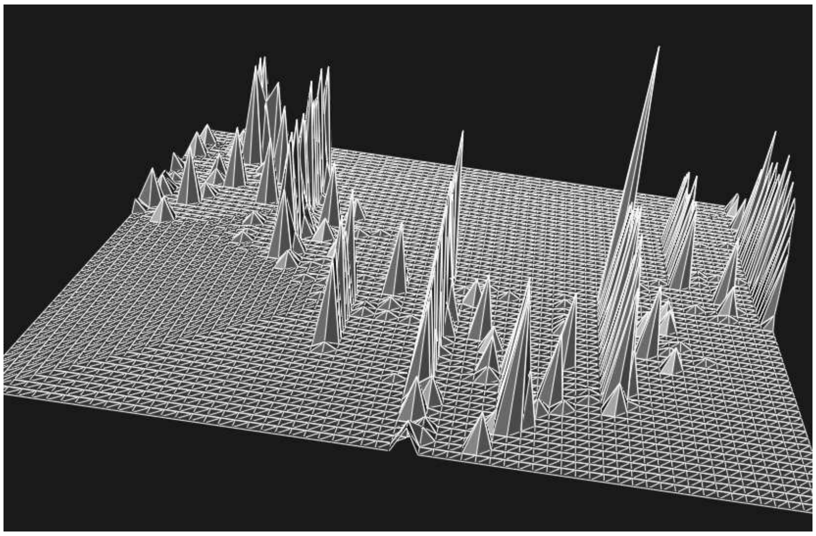

Figure 18: 3D dual space for *Irisa*

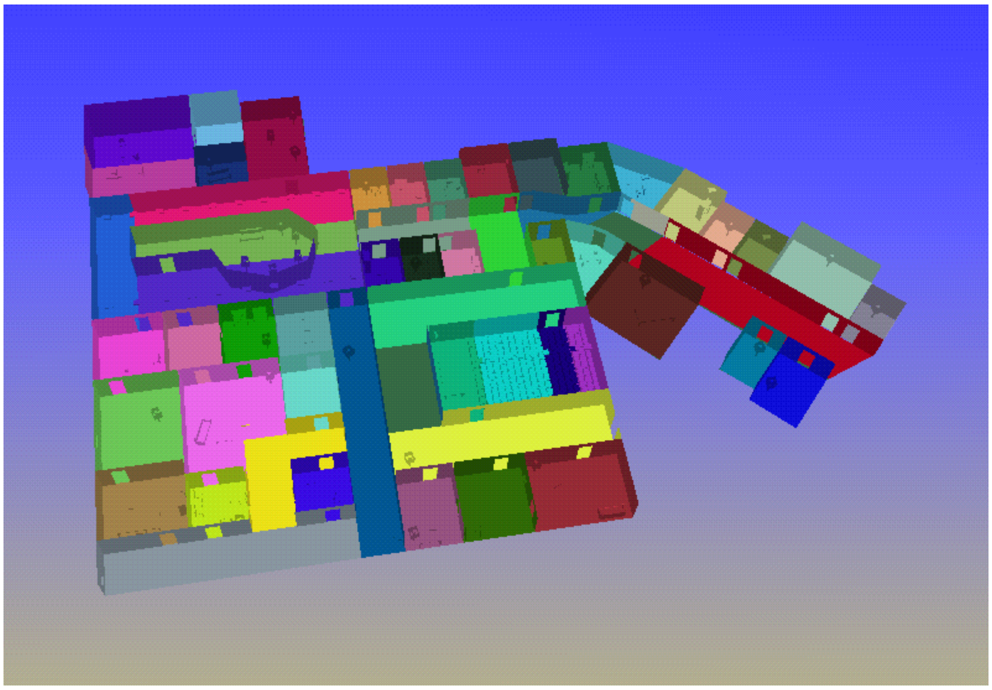

Figure 19: top-view of *Irisa*

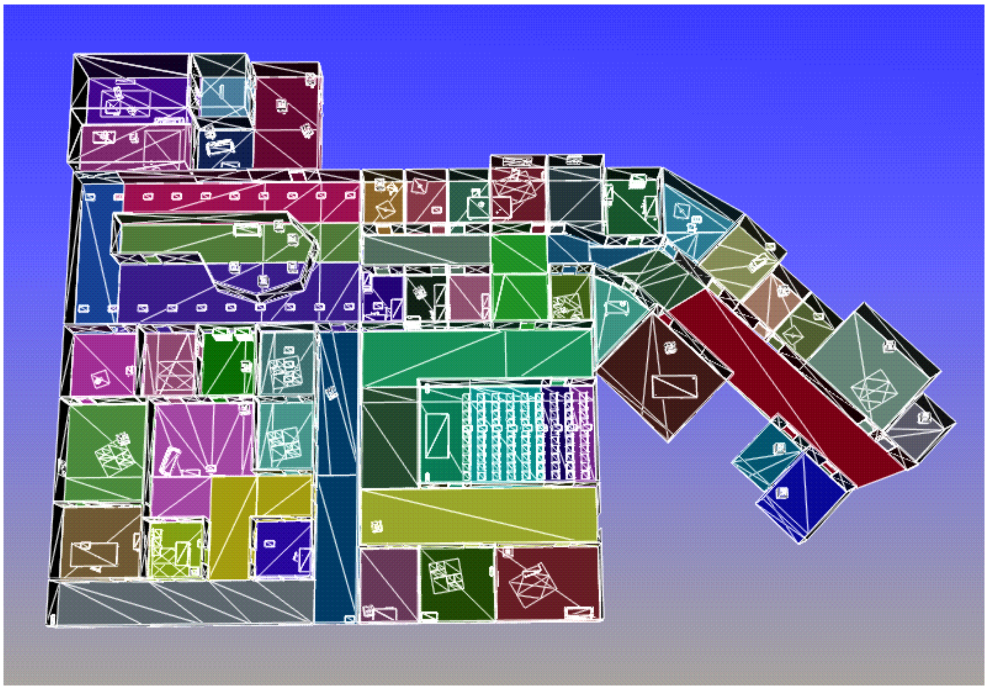

Figure 20: A wire-frame image of *Irisa*