A Spatial Representation for Ray-Scene Intersection Test Improvement in Complex Scenes

J. Revelles, N. Aguilera, J. Aguado, M. Lastra, R. García, R. Montes

Dpt. Lenguajes y Sistemas Informáticos, E.T.S.I. Informática, University of Granada, Spain
e-mail: [jrevelle,mlastral,ruben,rosana]@ugr.es
URL: http://giig.ugr.es

Abstract

We present a spatial representation based on a hierarchical structure using the well-known technique of octree as spatial indexing. There are very useful spatial representations for scenes whose objects can be distributed in clusters. Our proposed spatial representation is quite interesting for scenes that satisfy the previous objects distribution. In order to prove its benefits, several results are shown in scenes using a rendering algorithm to compute the global illumination based on photon map. These results show that the hierarchy of octrees becomes a good choice to improve it, taking into account the performances with other strategies such as octrees and hierarchy of uniform grids[3].

Keywords: Acceleration techniques, ray casting, spatial indexing methods.

1. Introduction

When a scene needs to be rendered, an algorithm based on ray-casting to compute the global illumination may be used. Mainly, these algorithms or methods will require a lot of time for ray-scene intersection test process. In order to improve the performance of this process, that is, to minimize the necessary time to compute the ray-scene intersection test, several techniques are available. These techniques are based on some kind of spatial indexing on the scene domain.

In this way, several spatial indexing techniques have been proposed as efficient strategies such that octrees[2, 5, 4, 11], 3D grid[1, 2], and BSPtree[9, 13, 8]. These spatial representations are even useful for different scenes, that is, objects with several sizes and with non-homogeneous distribution of them. In each space indexing a process which simulates a ray traversing this structure must be available. Therefore, Havran presented a good work comparing spatial representations and their algorithm to traverse them[7].

These spatial representations are proposed to accelerate the main process, that is, the ray-scene intersection test in a rendering system when a ray-casting based method is used. However, when the scene is very complex (many thousands of objects) both memory usage and rendering time could be too expensive. Another problem by the user is that not only on the rendering process but on the acceleration technique applied, several parameters must be fixed.

Due to the above reasons, a rendering system must contain several efficient techniques to accelerate the ray-scene intersection test. Moreover, the parameters required to build an specific spatial representation for a given scene must be easy to fix by the user. For this reason, in most of the cases a previous analysis for a given complex scene is required.

In this way, this paper shows the previous work in section 2. In this section several spatial representations which satisfy the above requirements are shown. In section 3 a new spatial representation is proposed. In order to show the acceleration technique performances which are proposed, several scenes have been rendered and analysed. A comparative timetable with the results are shown in section 4. In these results both memory usage and rendering time (illumination computation process) were considered. Finally, the conclusions and future works are presented.

2. Previous Work

There are two main aspects when a scene is rendered using a rendering system:

- Fix the necessary parameters in order to get a realistic image.
- For a given complex scene, fix an acceleration technique and the parameters required in order to increase its performances.

First step only depends on the rendering algorithm used. This problem may be considered on a second instance. The main problem is to select the best spatial representation to improve the most costly process when a scene is managed. When the mentioned spatial representation is selected, other problem derived from it is to look for the best values to initialize the parameters involved in it. In this case, several works were proposed[6, 12] for special cases of spatial representations.

In addition, a relevant contribution to solve this problem in complex scenes was introduced by Cazals et al.[3]. In that paper the authors proposed a new strategy to manage complex scenes in a rendering system which increased the performances of the ray-scene intersection test process. The proposed work was based on a regular grid and it was called hierarchy of uniform grids (*HUG*).

The main advantage of the strategy was because of distribution of objects in groups which are called clusters. In a first step, a filtering process of the input objects by size is done. Next, a clustering step to objects which are the same size is applied, and finally a hierarchy of regular grid is built for each cluster. In those terms, our spatial representation can be also used in these scenes because of the performances in runtime execution are increased.

The spatial representation which is proposed is based on the same philosophy of work proposed by Cazals. The unique different between the Cazals method and our method is that an octree is built for each cluster instead of a regular grid.

3. The Hierarchy of Octrees as Spatial Representation

We introduce a spatial representation which can be seen as a hierarchy of octrees (*HOO*). Using the same advantages of clustering process proposed by Cazals, the unique different is in the third step, that is, an octree is applied for each cluster.

The process to construct a hierarchy of octrees is described as follows:

1. The clustering algorithm[3] is used in a first step.
2. For each cluster, an octree is constructed.
3. Group two clusters neighbours into a bounding box (this process continues until a balanced binary tree representing the whole scene is obtained).

An example of a scene which a spatial representation based on a *HOO* may be applied is shown in figure 1. In this scene seven clusters will be obtained. For each cluster, an octree is built allocating all scene objects in this cluster (see figure 2).

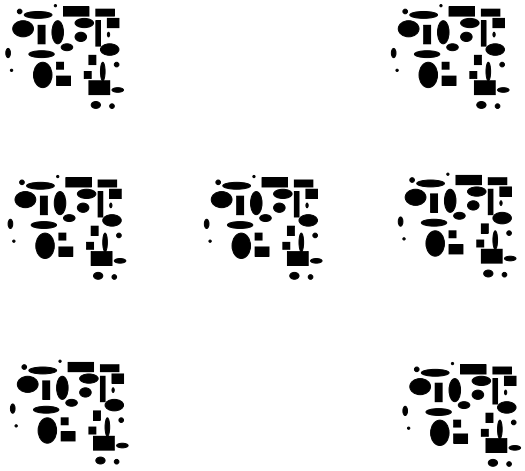

Figure 1: *Scene grouped in 7 clusters.*

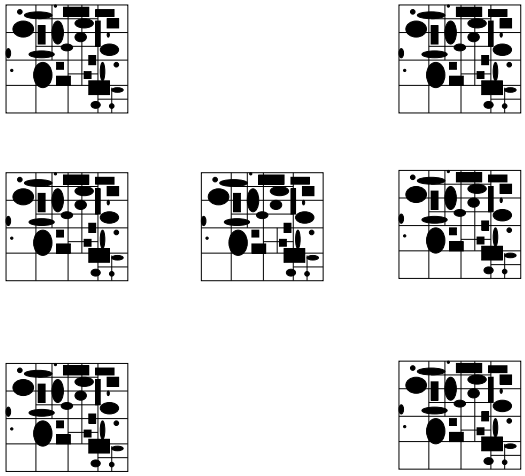

Figure 2: *Octrees applied for each cluster.*

In some cases, a scene can not be pierced into clusters because of the clustering algorithm do not find the existing number of clusters (in most cases, the whole scene could be matched as a unique cluster). In this situation, other considerations must be taken into account to use these spatial representations.

When the scene is feasible to split it in clusters, the following advantages are reached using a *HOO*:

- The spatial representation requires less memory usage than a simple spatial representation for the whole scene (i.e. using a regular grid or an octree).
- The performances increase when a greater space between

clusters occurs. These performances are obtained comparing *HOO* and an octree.
- Finally, a *HOO* provides better result than a *HUG* as it is shown in section 4.

The main disadvantage of a *HOO* or an *HUG* consists in the required time used in the clustering process. However, the clustering algorithm has a linear increase in relation to the number of objects.

4. Results

In order to show the performances of the proposed spatial representation, two comparisons and scenes have been proposed to do:

- Comparisons between *HOO* and octree (scene shown in figure 3). This scene is composed of two clusters. One of them is shown in figure 4).
- Comparisons between *HOO* and *HUG*.

All of results are obtained using a Pentium 4 processor with 1GB of RAM. The rendering algorithm is one based in global illumination by photon map[10]. The time considered is only related to illumination computation. The irradiance computation time is not taken into account. In the above

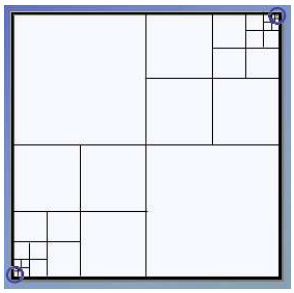

Figure 3: *Scene with two clusters and a greater space between them.*

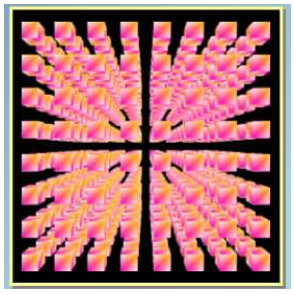

Figure 4: *Cluster of the above scene described as an array of* $8 \times 8 \times 8$ *cubes.*

scene, there are two clusters with a great division between them. Each clusters consists of an array of $8 \times 8 \times 8$ cubes with an extended light source over it. Each cube has 12 triangles (it will be the maximum number of objects allocated in a leaf node for an octree). The scene objects are triangles. In order to build an octree for one cluster considering 12 objects per leaf node, the maximum depth level of it will be 3. Nonetheless, when both clusters are considered to construct a unique octree, the maximum depth level would become 10 as it is considered in this case. Therefore, the memory requirements for a unique octree are larger than a *HOO* (see table 1). The gain of *HOO* is because of the necessary time

Octree		***HOO***
10	*Depth level*	3
1147	*Illumination Time (in seconds)*	139
	Gain percentage	87%

Table 1: *Octree vs. Hierarchy of Octrees.*

to traverse the internal nodes in the octrees until a leaf node is reached. In this case, it is very easy to determine the maximum depth level for both spatial representations. When this value is hard to fix, a good measure to compare the performances of them is when the memory amount in both spatial representations are similar. Therefore, a spatial representation based on a *HOO* has better results than an octree. The performances are greater when the clusters are more spaced out between them.

In order to compare *HUG* with *HOO*, a scene with 10 clusters is proposed (see figure 5). This scene has 50000

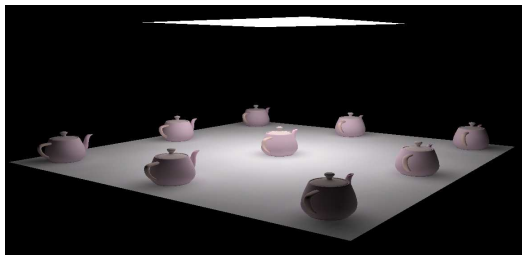

Figure 5: *Scene with 10 clusters (9 teapots and the floor).*

triangles including the floor. All the triangles have a similar size. In order to build a spatial representation based on a *HUG* or *HOO*, the number of clusters is 10. The rendering algorithm used is photon map with 1000000 photons per light source.

The results are shown in table 2. Octree depth level and subdivisions in regular grid are fixed to get spatial representations with similar memory usage in both cases. In these terms, the hierarchy based on an octrees presents better times than it is based on a regular grid.

HOO depth level	Time	%	Time	*HUG* Subdivisions
4	**64.55**	21%	**81.23**	13
5	**52.19**	35%	**80.10**	21
6	**48.73**	39%	**80.26**	29
7	**48.74**	39%	**80.35**	37
8	**49.35**	38%	**79.34**	45
9	**49.97**	38%	**80.51**	53

Table 2: *Hierarchy of Octrees vs. Hierarchy of Regular Grids.*

The gain in percentage displays results between 21% and 39% according to the times obtained in table 2.

At this point, it is feasible that when the input objects are quite greater and the rendering algorithm parameters are more accurate (in order to get images with the best quality) the gains will be similar.

5. Conclusions and Future Effort

In this paper we have introduced a spatial representation for very complex scenes. It is based on the same principles proposed in a previous work[3] but using octrees instead of regular grid. So, this work has similar advantages and disadvantages than *HUG*.

The main contribution of this spatial indexing consists of it has less memory requirements obtaining better gains in terms of execution time for the illumination computation process as it is displayed in the obtained results.

A good results must be obtained for cluster oriented scenes. The gain is increased when great distances between clusters exist.

As a future work, we will study the benefits of this spatial representation for more complex scenes and for different clusters distributions. In addition, we will study the importance of the space between clusters to study better the improvements that it would offer.

Acknowledgments

Special thanks to Carlos Ureña and Luis Miguel Vílchez for their contributions to this work. This work has been supported by a grant coded as TIC2001-2932-C03-03 (Spanish Commission for Science and Technology).

References

1. J. Amanatides and A. Woo. A fast voxel traversal algorithm for ray tracing. In *EUROGRAPHICS'87*, pages 3–10, Amsterdam, 1987.

2. J. Arvo and D. Kirk. *A Survey of Acceleration Techniques*, chapter chapter 6. An Introduction to Ray Tracing, pages 201–262. Academic Press, San Diego, 1989.

3. F. Cazals, G. Drettakis, and C. Puech. Filtering, clustering and hierarchy construction: A new solution for ray-tracing complex scenes. In *EUROGRAPHICS'95*, pages 371–382, 1995.

4. R. Endl and M. Sommer. Classification of ray-generators in uniform subdivisions and octrees for ray tracing. *Computer Graphics Forum*, 13(1):3–20, 1994.

5. A.S. Glassner. Space subdivision for fast ray tracing. *IEEE Computer Graphics & Applications*, 4(10):15–22, 1984.

6. J. Goldsmith and J. Salmon. Automatic creation of object hierarchies for ray tracing. *IEEE Computer Graphics & Applications*, 7(5):14–20, 1987.

7. V. Havran. *Heuristic Ray Shooting Algorithms*. Ph.d. thesis, Department of Computer Science and Engineering, Faculty of Electrical Engineering, Czech Technical University in Prague, 2001.

8. V. Havran and J. Bittner. Rectilinear bsp trees for preferred ray sets. In *Proceedings of 14th Spring Conference on Computer Graphics, Budmerice in Slovakia*, pages 171–179, 1999.

9. V. Havran, T. Kopal, J. Bittner, and J. Zara. Fast robust bsp tree traversal algorithm for ray tracing. *Journal of Graphics Tools*, 2(4):15–24, 1997.

10. M. Lastra, C. Ureña, J. Revelles, and R. Montes. A particle-path based method for monte carlo density estimation. In *Eurographics Workshop on Rendering (short paper) 2002, Pisa (Italy).*, 2002.

11. J. Revelles, C. Ureña, and M. Lastra. An efficient parametric algorithm for octree traversal. *Journal of WSCG (Copyright UNION Agency-Science Press)*, 8(2):212–219, 2000.

12. K.R. Subramanian and D.S. Fussell. Automatic termination criteria for ray tracing hierarchies. In *Proceedings of Graphics Interface'91*, pages 93–100, June 1991.

13. K. Sung. *Ray Tracing with the BSP Tree*, pages 271–274. Academic Press, 1992.