REPRESENTATION OF POLYHEDRAL OBJECTS USING *SP-OCTREES*

P. Cano, J.C. Torres

Department of Lenguajes y Sistemas Informáticos
University of Granada. E.T.S. Ingeniería Informática
Avda. Andalucía, 38. 18071 Granada
Spain
e-mail: [pcano, jctorres]@ugr.es
URL: http://giig.ugr.es

ABSTRACT

Extensions to classical Octrees that add new types of terminal nodes have been proposed for the exact representation of polyhedral objects. In this work, we present a new solid representation scheme using Octrees, which include boundary information of the represented object in the internal nodes of the tree. In this way, basic operations with the model will be accelerated and we are able to represent polyhedral objects exactly with smaller storage cost.

Keywords: Solid modelling, Geometric modelling, Hierarchical modelling, Octrees, BSP, Visualization, Polyhedra.

1. INTRODUCTION

Some of the schemes used to represent solids and volumes are based on the decomposition of the space, and use hierarchical structures to store the model. Two of them are the Octrees and the Binary Space Partition tree (BSP).

An Octree is the representation of a model by means of an octal tree structure obtained by recursive divisions of the bounding box of the volume to codify [Meagh82][Fujim84a][Fujim84b] [Garga82]. These divisions are made so that the octants obtained in each level of the decomposition are of equal size, repeating the process until the properties represented in all the nodes of the tree are homogenous or until arriving to an pre-defined level of depth.

The representation with Octrees is usually more concise than the space enumeration and allows to perform boolean operations and calculation of properties in a simple way, but it is an approximated representation of the solid.

To improve the classical Octrees, hierarchic schemes have been proposed that allow us to obtain an exact representation of polyhedral objects by means of the inclusion of new types of terminal nodes that contain part of the surface of the object, obtaining thus a more compact representation [Brune85][Brune90].

The Binary Space Partition trees (BSP) divide recursively the space using a plane in two separated semi-spaces. Initially created to improve the process of the elimination of hidden parts [Fuchs80] [Fuchs83], it has also been used to represent polyhedral objects exactly [Thiba87]. This scheme offers an unambiguous, but not unique representation, and allows us to perform boolean operations easily [Thiba87][Naylo90].

In this work an extension of the classical Octrees is proposed by means of the inclusion of information of the boundary of the solid represented not only in the terminal nodes, but also in the internal nodes of the tree. In this way, we do not always have to descend to the lowest level to accede to that information and we will be able to accelerate basic operations on the model.

In the following section the general statement of the new scheme and the data structure

used to store the model is presented, according to our proposal. In section 3 we describe the construction of a model in the proposed scheme from a B-Rep description of a solid. In section 4 we describe some of the basic operations on solids using the proposed scheme. Finally, the conclusions reached and work pending are presented.

2. SP-OCTREE

When trying to extend the Octree scheme representation, the modifications can be made in three different points of it:

a) By modifying the information included in the leaf nodes of the octal tree that represent the object. To do that, we can add information of the boundary of the object that appears in each terminal node [Brune85][Brune87] [Brune90].

b) By modifying the cutting planes used in the subdivision process [Cano96][Torre96] [Whang95].

c) By modifying the information stored in the internal nodes of the tree.

In classical Octrees and the extensions proposed, the internal nodes are those that are not homogeneous with respect to the classification criteria. So, in these nodes the only information appearing is the references to its children.

Extended Octrees [Brune85][Brune90] include information of the solid boundary in terminal nodes. So, the same boundary plane can appear in several neighbouring terminal nodes that share the boundary faces.

The idea of the proposed scheme, that we have called SP-Octrees (Space Partition Octrees), is based on the inclusion of boundary information in internal nodes that partially defines the object represented in each node of that level. Been more precise, we include the face planes that divide the voxel into an empty region and a partially occupied region. Thus, the information of the boundary faces appears in the upper levels of the tree and it is not necessary to repeat the information in neighbouring nodes that share a face.

When a node is completely in or out of the represented solid we classify it as BLACK or WHITE in the same way as in classical Octrees.

When the intersection of the solid and the voxel is concave, we use a CONVEX node. Formally, a CONVEX node is the intersection of the semi-spaces defined by the planes included in it with its bounding box. These nodes allow the exact and univocal representation of a convex polyhedral object.

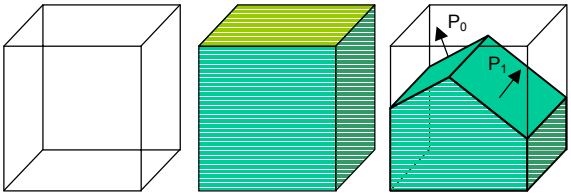

Figure 1. WHITE, BLACK and CONVEX nodes

When the intersection of the voxel and the solid is concave we use a CONCAVE node. Formally, a CONCAVE node is the difference of the bounding box of the node with the intersection of the complement of the semi-spaces included in it.

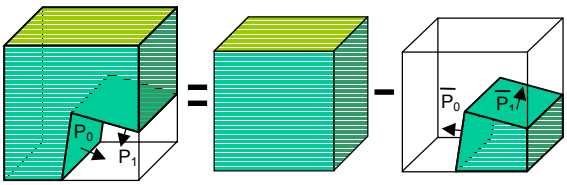

Figure 2. CONCAVE node

Finally, when concavities and convexities exist at the same voxel, we classify the node as GREY, dividing it in the same way as in classical Octrees, but maintaining in the node the information of the planes that are in the convex hull of the part of the solid in the node. Thus, in the children we only need to represent the boundary planes that are not in that convex hull and which form the existing concavities.

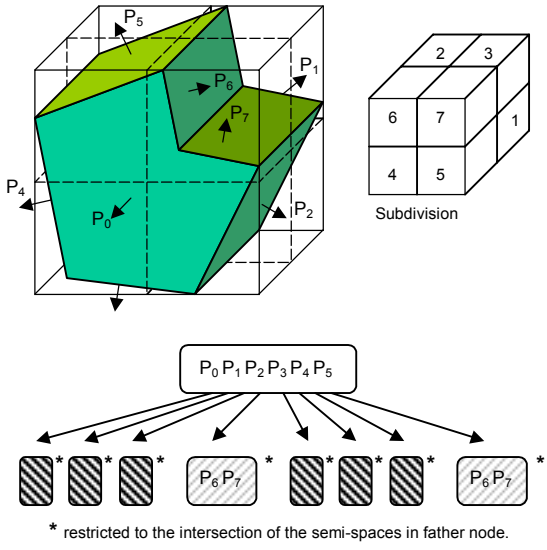

Figure 3. GREY node and its tree.

Thus, the solid represented by a GREY node will be the union of the solid represented by each child, but restricted to the intersection of the semi-spaces defined by the planes that appear in the father node with the bounding box of this node.

If P_k are the planes that define the existing semi-spaces in a node and *Box* is the bounding box of a node, we can define the representation R_S of a solid S by means of this scheme as:

$$R_S = \begin{cases} \left(\bigcup_{i=0}^{7} R_S^i \right) \bigcap_{k=0}^{P_i} P_k & (Grey) \\ \phi & (White) \\ Box & (Black) \\ Box \bigcap_{k=0}^{P_i} P_k & (Convex) \\ Box - \left(\bigcap_{k=0}^{P_i} \overline{P_k} \right) & (Concave) \end{cases}$$

where R_S^i is the representation of child i in the division of a grey node, and $\overline{P_k}$ are the complement of the semi-spaces defined by the plane P_k .

2.1. Proposed structure

In order to represent internally the proposed scheme, we will use the following representation:

- An auxiliary matrix with the equations of the planes used which define the semi-spaces whose intersection represents the solid in each node.
- The codification of the octal tree, where:

  a) In the terminal nodes, we store the type of node and a reference to the planes of the boundary included in it.
  b) In the internal nodes, we store the face planes that are in the convex hull of the solid represented in that node, plus links to the eight children that will contain the information of the planes that form concavities in the father node.

Any node includes planes that are included in any of its ascendant nodes. The BLACK nodes can be treated exactly like the CONVEX nodes where there are no planes of the boundary of the solid. The WHITE nodes have an empty set of planes.

The concave nodes will store the reference to the planes that form concavities in the solid. We must invert their normals when we have to process them, to treat them according to the definition that we have made in the previous section.

As we can see, the geometry of the represented object is not stored internally, which enables the representation obtained to be compact and reduces storage requirements.

3. BUILDING THE MODEL FROM B-REP

The construction of the proposed structure from a boundary representation of a polyhedral solid, is made by the following recursive algorithm:

1. Compute the bounding box of the solid, which will be the one of the root node of our representation.
2. Classify the node according to the planes included in it.
3. If the node is WHITE, CONVEX or CONCAVE, a terminal node with the information of those planes is created.
4. If the node is GREY, that is, if concavities and convexities at the boundary of the solid that it represents exist simultaneously, an internal node is created. This node includes the planes that are in the convex hull of this part of the solid, that are not included in any ascendants of this node. Then, we divide the node in eight equal octants, processing each one of them with the remaining planes that form the concavities in a recursive way (step 2).

The algorithm is repeated until all the nodes are classified, or until reaching a predefined maximum level of depth. In case of reaching that maximum level with a GREY node, we should establish some classification criteria. So, the representation obtained in those cases is approximated, and the error will depend on the level of the tree.

Image 1 shows a SP-Octree of level 0 for a convex object. Images 2, 3 and 4 show examples of SP-Octrees of level 1, 2 and 5 respectively, where the blue planes (darker) are those which pertain to planes in concave nodes.

Image 5 shows a SP-Octree of level 4 where the committed error when classifying as BLACK the GREY nodes of the maximum level (those nodes where concavities and convexities of the solid exist simultaneously) can be seen.

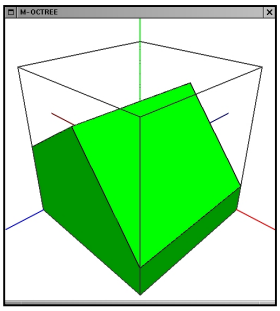

Image 1

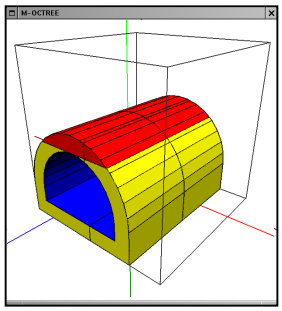

Image 2

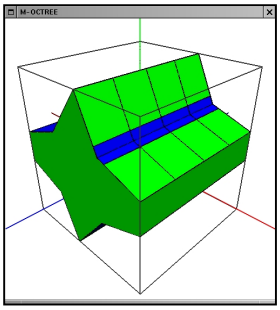

Image 3

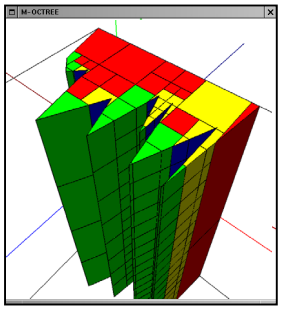

Image 4

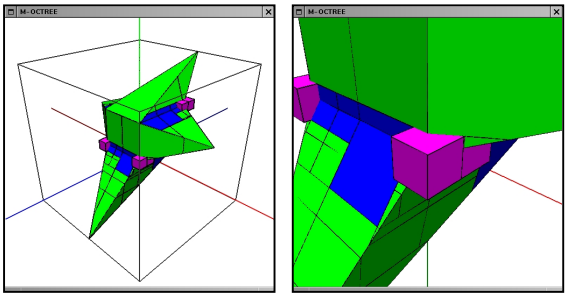

Image 5

4. BOOLEAN OPERATIONS

One of the main advantages of the Octrees is that boolean operations are simple and fast. In contrast, to make boolean operations between two B-Rep solids we must operate all the planes of a solid with those of the other to obtain the resultant boundary, with the computational cost that this entails. In our scheme, maintaining the hierarchical codification of the Octree, we reduce that cost since we are going to traverse the trees of both models at the same time, making the operation only between the planes that appear in each one of the corresponding nodes.

The representation of the complementary object can be obtained traversing the tree and changing the orientation of the planes that appear in each node. A WHITE node will be converted into a BLACK node and vice versa, while a CONCAVE node will be converted into a CONVEX node and vice-versa. When we process a GREY node, the convex planes that appear in it will pass to be concave and they should be propagated to the children, while the planes that represented the children nodes (and who belonged to concavities) will pass to the upper node as convex.

The process to make the intersection of two objects represented with the proposed scheme is based on the following rules:

- If one of the nodes is WHITE, the intersection will be a WHITE node.
- If one of them is BLACK, the intersection will be the copy of the other node.
- If one of them is CONVEX, we calculate if their planes are part of the convex hull of the result, computing their intersections with the planes in the other node, classifying the resultant node according to the existence or not of intersections between these planes.
- If one of them is CONCAVE, the process is similar to the previous one.
- If one of the nodes is GREY, we should operate the other node with the existing planes in the treated node and propagate the result upon its children (to trim them according to the new convex hull obtained in the father).
- If both nodes are GREY, we should calculate the intersection of the existing planes in each one and process the children propagating the information of the resulting planes in the father node obtained.

Images 6 and 7 show two examples of the intersection described. In each case we can see the models used and the result obtained.

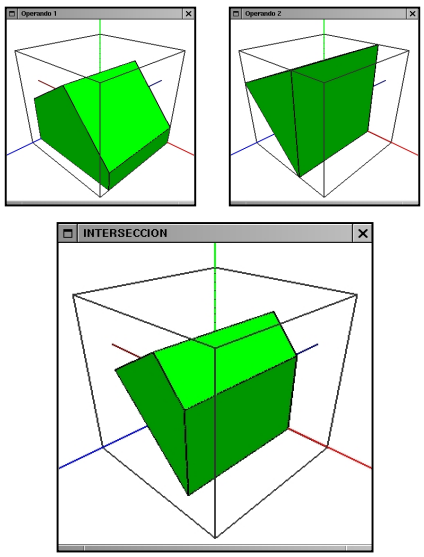

Image 6

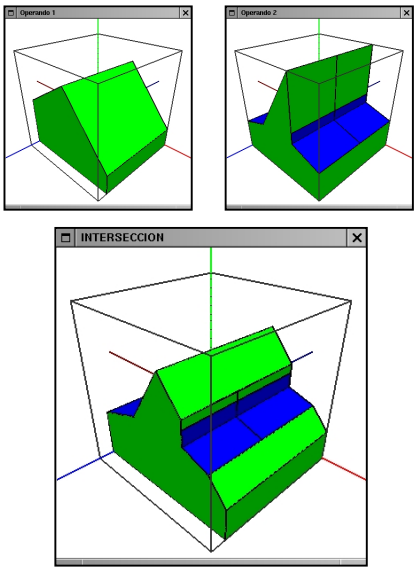

Image 7

The union and difference operations can be obtained by the same mechanism modifying the rules that define each one of the possible cases. Of course, we can also obtain them from the previously defined operations of complementation and intersection and the properties of the boolean operations.

As we are traversing the trees, the complexity of these operations depends on the number of levels and nodes in the operands trees.

5. RECONSTRUCTION OF THE BOUNDARY MODEL AND VISUALIZATION

Another of the advantages of the classical Octrees is the inherent arrangement in the scheme, which facilitates the visualization process defining the order of visualization of the nodes. In our case we continue maintaining that arrangement.

In order to visualise an object represented by means of the proposed scheme, we traverse the tree level by level, representing for each node the intersection of the planes that appear in it with its surrounding box and with the planes that appear in their ancestors. In this way, as we have information of the boundary of the object in the upper nodes, the higher levels of the tree allow us to obtain quickly the convex part of the boundary of the object. This mechanism allows us to make an adaptive visualization according to the level of the tree that we represent.

In order to draw the object faces it is necessary to trim the planes in one node against those in its descendants. We can do this while drawing or we can modify the data structure to store in each node of the tree, not only the definition of the planes, but also the geometry of the faces of the solid. We can obtain this easily using a secondary B-Rep scheme to accelerate the process.

Image 8 shows one object represented with classical Octrees (top), Extended Octrees (middle) and the proposed scheme SP-Octrees (bottom). In these images we can observe the approximation that the classical octrees generates, while the other two schemes represent the objects exactly.

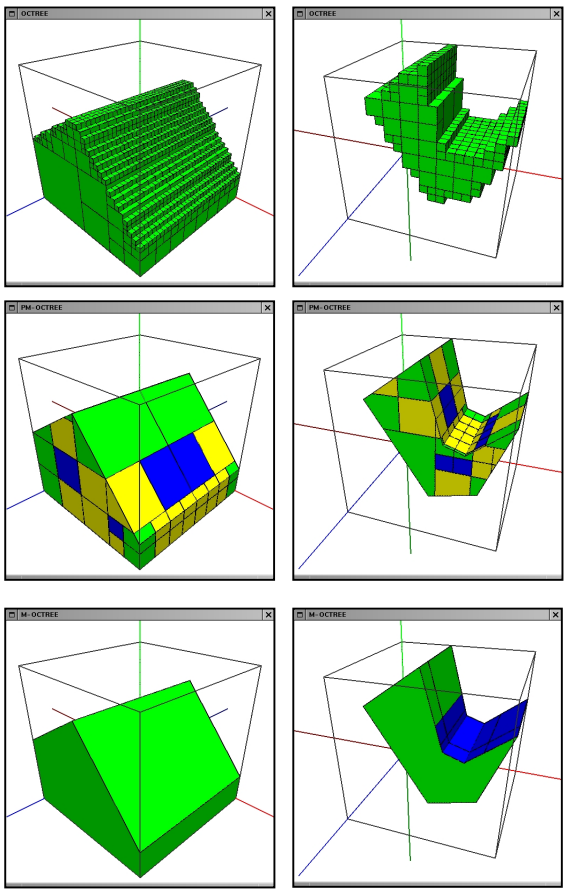

Image 8

Using the process described to visualise the objects represented with the proposed scheme we can reconstruct the boundary of the solid.

We must notice in this process the possibility that some of the faces of the solid appears divided in various polygons when being shared by various nodes. In this case, we must do a post-process of the obtained faces in order to merge those faces that are in the same plane. This will reduce the number of polygons used to represent the obtained boundary.

Finally, we must consider that, as the representation can be approximated in the cases in

which the maximum level of depth defined is reached in a node with concavities and convexities (section 3), the obtained boundary in that case will be an approximation of the real boundary of the solid.

6. CONCLUSIONS AND FUTURE WORKS

In this work a new solid representation scheme has been presented based on an extension of the concept of classical Octree, introducing part of the boundary information of the represented object, both in the terminal and in the internal nodes.

The proposed method allows an exact representation of polyhedral objects except when vertex shared by concave and convex edges exist (see image 5). In that case, we will always reach the maximum defined level for the tree and will have to establish a criterion of classification for that node, what make the representation to be approximate.

The number of nodes that appear in the presented scheme depends only on the concave edges that appear in the solid, whereas in the Extended Octrees the number of nodes depends on the number of vertex and edges of it.

Because we can store the information of the planes that define the boundary of the solid in an auxiliary structure, and we reduce the repetition of planes in neighbouring nodes that shares them inserting them in inner nodes of the structure, the need of space is smaller than with Extended Octrees.

In addition, we continue maintaining the properties of arrangement of the classical Octrees, and, due to the own orientation of the planes inserted in each node, it is easy to make the interrogation and visualization of the model.

Now, we are studying the cases that cause this scheme to be approximated to try to define some method that allows us the exact representation of any polyhedral object. This solution will probably use the insertion of auxiliary planes that divide the node in a set of disjoint areas in which we can use the proposed scheme.

We are making a detailed comparative study with other representation schemes, both in space and computation time and operations complexity. Also, we are studying the possibility of using the scheme for objects whose boundary is not plane.

Finally, we are studying the utility of the scheme as an indexing method to accelerate the calculations and the operations in B-Rep representation scheme, as well as its use in the transmission, visualization and progressive edition of the represented models.

7. ACKNOWLEDGEMENTS

This work has been supported by the *"Comisión Interministerial de Ciencia y Tecnología" (CICYT, Spain)* under contract TIC2001-2099-C03-02.

REFERENCES

[Brune85] Brunet, P.; Navazo, I.: *Geometric modelling using exact octree representation of polyhedral objects.* Eurographics'85, 1985.

[Brune87] Brunet, P.; Ayala, D.; *Extended octree representation of free form surfaces.* Computer Aided Geometric Design, 4, pp: 141-154, (1987).

[Brune90] Brunet, P.; Navazo, I.: *Solid Representation and Operation Using Extended Octrees.* ACM Transactions on Graphics, Vol. 9, nº 2, pp: 170-197, (1990).

[Cano96] Cano, P; Velasco, F.; Torres, J.C.: *Modelado 2D mediante jerarquía de Quadtrees en distintos sistemas de coordenadas.* II Jornadas de Informática, pp: 153-162, (1996).

[Fuchs80] Fuchs, H.; Kedem, Z.; Naylor, B.: *On Visible Surface Generation by a Priori Tree Structures.* ACM Computer Graphics, 14(3), pp: 124-133, (1980).

[Fuchs83] Fuchs, H.; Abram, G.D.; Grant, E.D.: *Near Real-Time Shaded Display of Rigid Objects.* ACM Computer Graphics, 17(3), pp: 65-72, (1983).

[Fujim84a] Fujimura, K.; Yamaguchi, K.; Kunii, T.: *Octree-related data structures ans algorithms*. IEEE Computer Graphics and Applications, pp:53-59, (1984).

[Fujim84b] Fujimura, K.; Toriya, M.; Yamaguchi, K.; Kunii, T.: *Octree Algorithms for Solid Modelling*. IEEE Computer Graphics and Applications, 1984.

[Garga82] Gargantini, I.: *Linear octrees for fast processing of three-dimensional objects.* Computer Graphics and Image Processing, 20, (1982).

[Meagh82] Meagher, D.: *Geometric modelling using octree encoding.* Computer Graphics and Image Processing, 19(2):129-147, (1982).

[Naylo90] Naylor, N.; Amanatides, J.; Thibault, W.: *Merging BSP Trees Yields Polyhedral Set Operations.* ACM Computer Graphics, 24(4), pp: 115-124, (1990).

[Thiba87] Thibault, W.C.; Naylor, B.: *Set Operations on Polyhedra Using Binary Space Partitioning Trees.* ACM Computer Graphics, 21(4), pp: 153-162, (1987).

[Torre96] Torres, J.C.; Cano, P.; Velasco, F.; Conde, F. *Using octrees in non cartesian coordinate systems.* Internal Report LSI-96-1, (1996).

[Whang95] Whang, K.Y.; et al.: *Octree-R: An Adaptative Octree for Efficient Ray Tracing*. IEEE Transactions on Visualization and Computer Graphics, Vol. 1, nº 4, pp:343-349, (1995).