

# Multiresolution Isosurface Fitting on a Surface Octree

Imma Boada\* and Isabel Navazo†

\* Institut Informàtica i Aplicacions, UdG, Spain

† Dep. Llenguatges i Sistemes Informàtics, UPC, Spain

## Abstract

The Surface Octree (SO) is an extension of the Classical Octree used to maintain a decimated codification of a surface while preserving volume data information. Constrained by a region of maximal interest, we present an algorithm to extract a multiresolution surface from a SO data structure. This algorithm reconstructs the surface with maximal precision in the focus of interest and at progressively lower resolution away from it.

## 1 Introduction

The work we are presenting here is part of the design of an *hybrid model* [2]. This model, which is based on an octree scheme, is able to maintain and efficiently manipulate surface and volume data, either *independently*, taking advantage of classical surface and volume visualization approaches, or *integrated*, combining surface and volume data in a single image. In particular, we are going to present part of the work related to the surface manipulation.

Since the earliest work from Meagher [7] concerning 3D data representation based on the octree scheme, many applications of this hierarchical representation have appeared over the years, e.g., to improve rendering speed, to manage easily large data sets, or to obtain error controlled renderings [4, 5, 16, 17]. In surface rendering applications, they have been used in combination with the Marching Cubes (MC) algorithm [6]: to accelerate the surface extraction process or combined with adaptive isosurface extraction techniques as a *surface simplification method* [13, 11, 15]. In this last case, the octree is used to subsample the data and extract isosurface patches just applying the MC to those nodes which meet certain criterion.

The Surface Octree (SO) is an extension of the classical octree used to maintain a decimated codification of an isosurface [1]. Additionally to the classical black, white and grey nodes, the SO introduces five new terminal nodes. The most remarkable properties of the SO are that:

- it is able to maintain a codification of any surface obtained from the MC algorithm.
- it reduces the number of faces of the fitted surface without introducing any error [1].
- it provides a framework able to support multiresolution surface renderings.
- it preserves the relation between the fitted surface and the volume data. Thus, it is a suitable model for supporting multiresolution hybrid surface and volume renderings. For our purposes, this is the most important characteristic.

In this paper we propose an algorithm to adaptively extract a surface from a SO data structure. Constrained by a region of maximal interest (ROI), the algorithm reconstructs the surface with maximal precision in this area and away from it at progressively lower resolution.

The paper is organized as follows. The SO data structure and its construction process are briefly revised in section 2. Subsequently, in section 3, we describe how to extract a surface from the SO without introducing any error. In Section 4, we analyze the proposed surface simplification strategy and the implemented crack patching technique. The multiresolution surface fitting algorithm is presented in Section 5. In Section 6, several results achieved with the proposed technique show that it is well-suited to perform interactive navigations with complex fitted surfaces. Finally, concluding remarks and future work are presented in Section 7.

---

\* imma@ima.udg.es

† isabel@lsi.upc.es

## 2 The Surface Octree

It is well-known that an octree is a tree that codes a finite cubic universe by its recursive subdivision. The root of the tree represents the universe, a node with  $2^n$  edge. This node is divided into eight identical cubes, called octants, with an edge length of  $2^{n-1}$ . In this structure each node is terminal or has eight descendants. In the classical octree representation [12] nodes are labeled as black, white and grey. *Black* nodes correspond to regions completely inside the object. *White* nodes correspond to regions completely outside the object. White and Black nodes are leaves, i.e. they are no further subdivided. Nodes containing part of the surface are labeled as *Grey* and are recursively subdivided. Leaf grey nodes are named *Terminal Grey*.

In the recent years, several extensions of this octree model have been proposed. Such extensions introduce terminal configurations to be added to terminal Black, White and Terminal Grey nodes. *Face Octrees* [14], for example, introduce the Face Node (which contain a face of the surface of the the object being modeled) or *Extended Octrees* [10], that introduce *Face nodes* (crossed by a single planar face of the solid), *Edge nodes* (contain part of two neighboring faces and a part of their common edge), *Vertex nodes* (contain a vertex of the polyhedron and a part of the faces and edges converging to it) and *Nearly Vertex nodes* (contain two or more faces that converge to a single vertex outside the node).

### 2.1 Terminal Surface Nodes

In order to obtain a well-suited octree to codify a surface extracted by the MC algorithm, the SO introduces to the classical Black and White nodes, five new Terminal Surface (TS) nodes (see figure 1). These nodes are characterized by the topology and the number of polygons that define the surface contained in them. These five TS configurations are:

- **Face nodes**, ( $F$ ), crossed by a single planar face.
- **Edge nodes**, ( $E$ ), crossed by two planar faces that converge into the node, generating an edge.
- **Double Edge nodes**, ( $DE$ ), crossed by three planar faces that converge into the node, generating two edges.

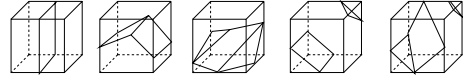


Figure 1: Terminal Surface nodes added to the classical octree.

- **Band<sup>n</sup> nodes**, ( $B^n$ ) crossed by  $n$  planar faces that never intersect into the node, with  $2 \leq n \leq 4$ .
- **Special Band nodes**, ( $B^*$ ), obtained from the union of a  $F$  and an  $E$  node.

The surface of a TS node is represented as  $s(n_i) = \{K, \langle V_0, d_0 \rangle, \dots, \langle V_n, d_n \rangle\}$  where  $K$  identifies the node's configuration (i.e.  $F$ ,  $E$ ,  $DE$ ,  $B^n$  or  $B^*$ ) and  $(\langle V_0, d_0 \rangle, \langle V_1, d_1 \rangle, \dots, \langle V_n, d_n \rangle)$  represents the set of plane equations that define the surface. Each couple  $\langle V_i, d_i \rangle$  corresponds to a  $\pi_i$  plane where  $V_i = (A_i, B_i, C_i)$  is the normal vector and  $d$  is the distance from the plane to the node's origin.

### 2.2 The SO construction

To obtain a SO, we start with a complete volume octree in which the isosurface, initially obtained with the DiscMC, is codified at terminal nodes according to the new configurations. *Note that using the DiscMC the number of planes incidences is reduced, and all the codification process is simplified. The surface codification can be reduced to a set of integers.*

Then, a bottom-up strategy is taken in merging cells. Such strategy allows to implicitly represent coplanar faces in upper octree levels without introducing error. As all possible TS node configurations are known, a look-up Merging Table (MT) is used to maintain all possible combinations of TS nodes and the way they have to be merged. The MT table is the basis of all the simplification process. The merging process ends with the surface codified in a set of TS nodes distributed at different levels of the octree.

## 3 Maximal Accuracy Surface Reconstruction

When an explicit representation of the fitted surface is needed, for rendering purposes for example, a surface reconstruction algorithm has to be applied

on the SO. This algorithm is composed of two steps. The first, traverses in a top-down manner the octree and selects all TS nodes. The second, reconstructs the surface contained in each one of the selected nodes. This reconstruction is carried out by the `ExactRec` function described in next section.

### 3.1 ExactRec Function

Before the description of this function we have to take into account that: (i) each plane equation represented in the TS node has to be represented by one polygon; (ii) this polygon is defined from the connection of *all* or *some* of the intersection points between the plane and the edges of the node. (iii) the number of polygons required to define the surface varies with the configuration of the node (F:1, E:2, DE:3, B<sup>n</sup>:n and B<sup>\*</sup>:3).

Let consider that the surface of a  $n_i$  node has to be reconstructed. The `ExactRec` function applies the three following steps:

1. *Intersection Point Computation.* On a first step  $s(n_i)$  is evaluated and for each plane equation  $\langle V_i, d_i \rangle \in s(n_i)$ , the intersection points between the plane and the edges of the node are computed. This set of points is represented as  $P_i$ . Additionally, for Edge, a DoubleEdge or a *Band\** nodes, the points defining the interior edges are also computed (see Figure 2(a)).
2. *Point Selection.* For each set  $P_i$  a selection process is applied to determine which of these points are members of the surface. Such selection depends of the node's configuration and is done evaluating the normal vector (see Figure 2(b)). Selected vertices are represented in  $P_i^*$ .
3. *Points Connection.* Finally,  $P_i^*$  vertices are connected using a look-up table.

The error involved in all the surface extraction process is the error introduced by the `DiscMC`, i.e. an error of 1/2 of the cell edge size.

## 4 Simplified Surface Reconstruction

The main goal of a simplified surface reconstruction is to reduce the number of polygons used to represent the surface while preserving the surface connection (no cracks must be generated). In order to do that, we propose a simplification technique for each class of SO node and an adaptive crack patching strategy.

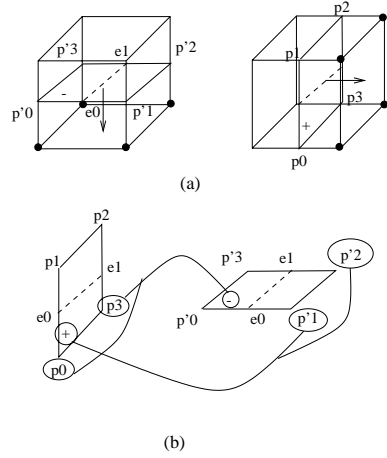


Figure 2: The surface of the node is codified as  $s(n_i) = \{E, - \langle 0, 1, 0 \rangle, 0.5, \langle 1, 0, 0 \rangle, 0.5\}$ . (a) Intersection Point Computation; (b) Point Selection according to the normal vector direction

### 4.1 Simplification Strategy

The vertices of the polygons of the surface inside a node could be classified as (see figure 3): *external* if they lie on one of the edges of the node or *internal* if they are not external.

Based on this vertices classification the simplification strategy eliminates all internal vertices and generates the polygons connecting the external ones. The critical point of this policy is to determine how external vertices must be connected. To solve this problem we have analyzed the possible situations and a specific solution is proposed for each case.

#### Simplification of a TS node

When the simplification has to be applied to a TS node,  $(n_i)$ . the surface can be obtained directly if one of the following conditions is satisfied:

- $n_i$  is an E node with the edge *isothetic* (i.e. the generated edge is parallel to one of the axis) (see Figure 3(a)). In this case  $n_i$  can be reduced to a face.
- $n_i$  is E node with the edge no isothetic. In this case, external points have to be detected and the two planes required to connect them have to be computed (see Figure 3(b)). The number of planes is not reduced. There is a reduction

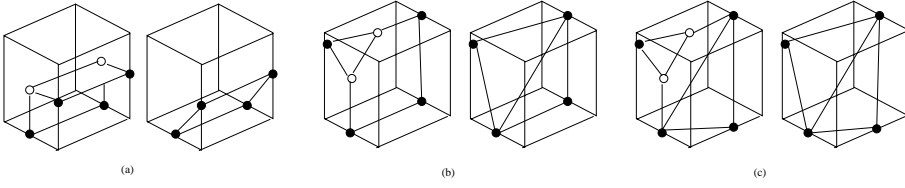


Figure 3: Black and white circles corresponds to external and internal intersection points respectively. (a) An Edge Node is reduced to a Face. (b) An Edge Node is represented as an Edge with a simplification on the number of surface vertices. (c) A DoubleEdge Node is simplified as an Edge.

on the number of intersection points.

- $n_i$  is a DE node, if the vertices of the generated edges are internal, the node can be reduced to an E node. Once the E is obtained, if it is isothetic it is reduced to a F (see Figure 3(c)).
- $n_i$  is a  $B^*$  node, with  $s(n_i) = \{B^*, \pi_0, \pi_1, \pi_2\}$ , and where  $\pi_0$  corresponds to the Face plane and  $\pi_1, \pi_2$  to the Edge. The simplification is applied on the Edge configuration as it has been previously described. In the optimal situation the new configuration corresponds to a  $B^2$  node.

If none of the previous situations is given, a selection process identifies the set of external vertices of the TS node. Then, a triangulation scheme is applied to connect them and generate the surface (see section 4.3 ).

### Simplification of a not TS node

When the node  $n_i$  is not a TS node, in order to detect the set of external vertices of  $n_i$  the TS descendant nodes are selected. Then, a triangulation scheme is applied to properly connect them (see section 4.3 ).

## 4.2 Crack Patching

A crack is a discontinuity that appears on common faces between neighbor nodes, when the intersection points defining the surface do not match. Such discontinuity generates a hole on the final surface. To describe the patching strategy we denote the face containing a crack as the *crack face*, and the points causing the crack as *crack points* (see Figure 4(a)).

Our proposed crack patching strategy varies according the relation between the crack face and the user defined ROI. We classify the nodes as IN and OUT, according to its relation with the ROI. An IN

node is a node inside the ROI and it must be represented with maximal accuracy. An OUT node is external to the ROI and it should have a simplified surface reconstruction.

Observe that a crack face only could appear in two situations:

- (i) when it is the common face between a node represented at maximal accuracy (TS node) and a simplified node. In this case, the crack is consequence of the elimination of internal intersection vertices performed by the simplification strategy;
- (ii) when it is the common face between two nodes that have been simplified and represented at different levels.

Based on the IN/OUT nodes classification the crack patching is done as follows:

- if the crack appears on the common face between an IN and an OUT node, the patching strategy is applied during the reconstruction of the OUT node. Crack points are considered as surface vertices. The coarser representation (OUT node) is forced to match with the higher one (IN node) (see Figure 4 (b)(c)).
- if the crack appears on the common face between two OUT nodes, the higher representation is forced to match with the coarser one, i.e. the points causing the crack are not considered on the surface representation process. (see Figure 4(b)(c))

This strategy guarantees that IN nodes are always reconstructed with maximal accuracy.

## 4.3 SimplRec Function

Summarizing, taking into account the simplification and crack patching strategies that have been defined, we propose the `SimplRec` function to reconstruct the surface of an intersected node  $n_i$  of

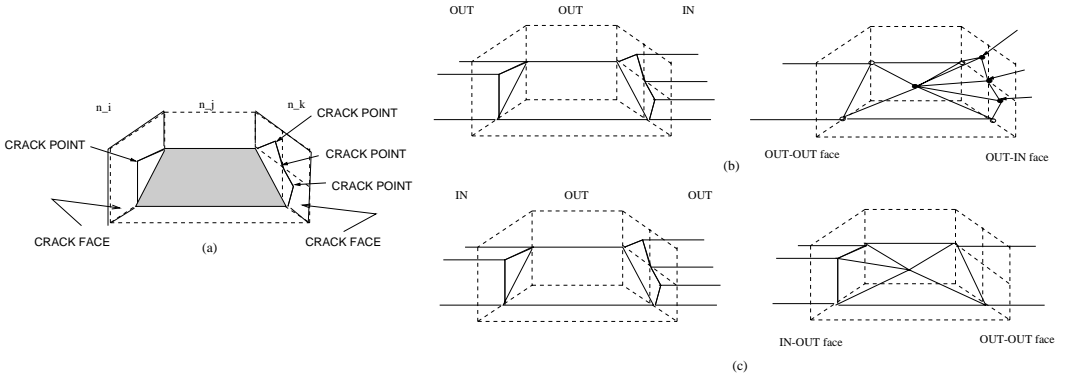


Figure 4: (a) Cracks faces and crack points of node  $n_j$ . (b) Patching strategy when the classification of  $n_i$ ,  $n_j$  and  $n_k$  nodes respect to the ROI is OUT, OUT and IN, respectively. (c) Patching strategy when the classification of  $n_i$ ,  $n_j$  and  $n_k$  nodes is IN, OUT and OUT, respectively.

level  $l$ . This function is composed of the following steps:

1. *External Vertices Detection.* Constrained by the simplification strategy, the surface only has to connect external vertices. Thus, the first step is the detection of the  $n_i$  external vertices. Such selection varies according the configuration of the  $n_i$  node:
  - (i) If  $n_i$  is a TS node and its original configuration allows a simplification (because some of the conditions defined in section 4.1 are satisfied) the set of intersection points defining the surface are obtained directly. Then, the Point Computation Strategy and the Point Selection process defined in section 4.1 are applied.
  - (ii) If  $n_i$  is not a TS node or is a TS node that could not be simplified, the external vertices are selected on a SO traversal.  $n_i$  TS descendants are localized and, subsequently its external vertices are identified. The set of external vertices is denoted as  $P^*$
2. *Crack Test.* The Crack Test compares vertices polygons obtained from a node and from its neighbors. When there's no matching between all the points crack faces are identified. According the situation of the crack face with respect to the ROI, the crack points are considered members of  $P^*$  or not.
3. *Points Triangulation.* Finally, the set of  $P_i^*$  points are triangulated. A look-up-table have

been defined to proper counter clock-wise ordering them.

## 5 The Multiresolution Isosurface Fitting Algorithm

In this section we present the algorithm to adaptively extract an isosurface from the SO data structure. The algorithm uses the reconstruction functions defined in the previous section, *ExactRec* and *SimplRec*.

The user-defined ROI is a simple 3D subregion of the dataset domain. The algorithm detects and reconstructs the nodes of the ROI, and then applies a simplification and crack patching strategies to generate the surface outside the ROI. Let analyze both steps in detail.

### 5.1 ROI reconstruction

The algorithm starts with the identification of intersected nodes of the ROI. These nodes compose what we call the  $C_{in}$ .

Starting at the coarsest level, SO nodes are recursively processed and

- if the region covered by the  $n_i$  node is completely internal to the ROI, the node is classified as IN. The  $n_i$  TS descendant nodes are detected and considered members of  $C_{in}$ .
- if the region covered by the  $n_i$  node is completely external to the ROI, the node is classi-

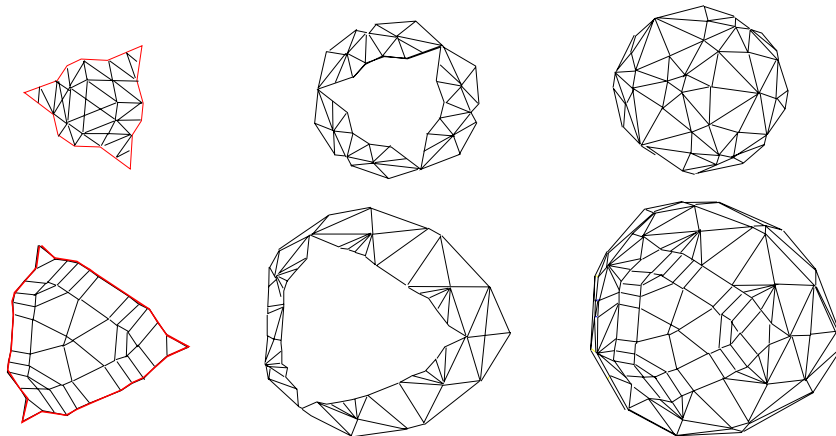


Figure 5: The first steps of the reconstruction process applied to two spherical models

fied as OUT. No more of its descendants will be processed.

- if none of the previous situations is given  $n_i$  descendants are evaluated

This process ends when all  $C_{in}$  nodes have been selected. To improve the SO traversal maximal and minimal values of the region covered by the node are stored [16].

To reconstruct the surface of the ROI, for each  $n_i \in C_{in}$  the *ExactRec* function is applied.

## 5.2 Outer ROI Reconstruction

In the next phase, the surface outside the ROI is reconstructed. The process applied for such reconstruction is represented in the following algorithm:

```

 $C_i = C_{in}$ 
Repeat
  DetectBorder( $C_i, C_{out}$ );
  for each  $n_i \in C_{out}$ 
    SimplRec( $n_i$ );
   $C_i = C_{out}$ ;
until  $C_{out} = NULL$ 

```

where *DetectBorder* (IN:  $C_I$ , OUT:  $C_O$ ) is a function that given a set of nodes  $C_I$  as input returns the set of nodes  $C_O$ , such that each  $n_i \in C_O$  satisfies: (i)  $n_i \notin C_I$  (i.e. the surface of the  $n_i$  has not been reconstructed); (ii)  $n_i$  is an intersected node; (iii) exists one  $n_j \in C_i$  18-connected (i.e.  $n_i$  is connected with a reconstructed node).

The set of nodes returned by this function is what we call *border nodes of  $C_I$* .

The process starts with the nodes of the ROI as input of *DetectBorder*. The function returns the border nodes that haven't been reconstructed. These nodes are reconstructed in a simplified manner. Then, border nodes of this nodes are detected and reconstructed. The process is repeated until *DetectBorder* returns the NULL list, this situation is given when all the surface is reconstructed.

In figure 5 we have illustrated the three first steps of the reconstruction process applied to two spherical models: first the region of maximal interest is reconstructed, then border nodes of the ROI and then the border of the ROI's border.

## 6 Experimental Results

The proposed algorithm has been implemented on a standard PC Pentium III at 450MHz with a NVidia graphic card. Three data models have been used for the tests: a CT-vertebra of 128x128x80; a CT-scanned jaw of 256x256x40 and a sphere of 64<sup>3</sup>.

First, we have considered the extremal situation, in which all the volume has been considered of maximal interest. In this case, the proposed algorithm works as a *classical* surface decimation method. The achieved reductions, on the number of polygons if compared with the MC algorithm, are: a 48%, in the best situation, the CT-vertebra (see

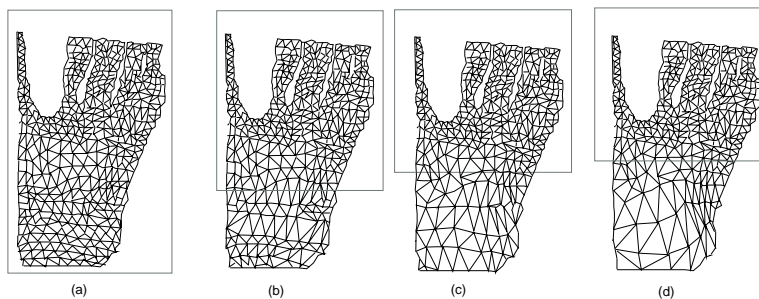


Figure 6: A lateral view of the CT-jaw modifying the ROI.

Figure 8); a 38% on the CT-jaw (see Figure 7); and a 35% on the  $64^3$  sphere. Timing results, including all the phases of the process, (surface octree construction, merging process and surface extraction), are: 4.5, 6.3 and 2.4 seconds for the CT-vertebra, CT-jaw and the sphere respectively.

The  $64^3$  sphere, has been used to compare the proposed algorithm with the DiscMC algorithm [9] and with Yagel's decimation method [13]. Best results are obtained with the DiscMC, with a degree of simplification of 76% with respect the MC algorithm. In our method the degree of simplification is of 35%, the same obtained by Yagel's. Note that, differently of DiscMC, we are forced to apply the simplification on the regular regions defined by the nodes of the octree. However, although our degree of simplification is lower, our method obtains triangulated representation that should possess a more regular shape than the ones produced by the DiscMC, which usually are composed of very thin triangles.

In Figure 6 a multiresolution reconstruction of the CT-jaw is shown. The ROI has been modified. The degree of simplification on the number of polygons, if compared with Figure 6(a) with 263.252 faces, are of 30% in figure 6(b), 45% in figure 6(c) and 54% in figure 6(d). When changing the ROI, previous computations can be used, i.e. not all the tree nodes have to be recalculated. Thus, the technique is well-suited to perform interactive navigations with very complex surface where the user could modify on-line the ROI.

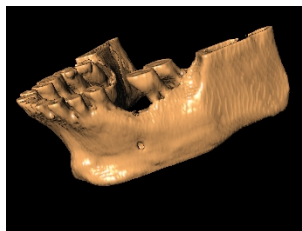


Figure 7: The CT-jaw model

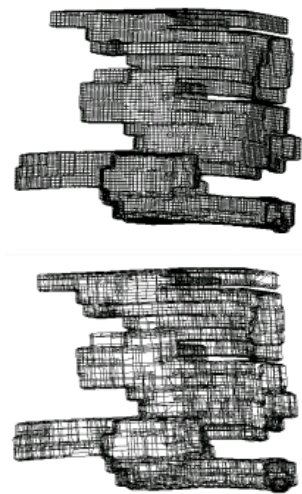


Figure 8: A CT-vertebra reconstructed with the Marching Cubes and extracted from the Surface Octree.

## 7 Conclusions and Future Work

We have presented an algorithm to adaptively extract a surface from an SO data structure. The SO is an extension of the classical octree used to maintain a decimated codification of a surface in a set of terminal nodes distributed at different levels of the octree. The surface is reconstructed from a set of nodes selected according its distance to a region of interest. The algorithm reconstructs the surface of maximal interest nodes with maximal accuracy. Non interesting nodes are reconstructed in a simplified manner. To guarantee the correctness of the reconstructed surface a crack patching strategy has been defined.

In spite of the fact that implementations could be further optimized, the experiments show the efficiency of the proposed multiresolution method.

The work presented here is part of the hybrid framework described in [2]. The connection between surface and volume data preserved in the SO data is exploited to support multiresolution surface and volume data renderings. Our future work will be centered on the definition of a multiresolution hybrid renderer that extends the work presented in [3].

## References

- [1] I.Boada and I.Navazo. An Octree Isosurface Codification based on Discrete Planes. Proceedings of Spring Conference on Computer Graphics 2001. pp.187-194. Budmerice,Slovakia,February 2001.
- [2] I.Boada and I.Navazo. The Hybrid Octree: a Multiresolution Surface-Volume Framework. Technical Report (IliA 01-09-RR). Girona University. 2001
- [3] I.Boada, I.Navazo and R.Scopigno. Multiresolution Volume Visualization with Texture-based Octree. The Visual Computer, Springer International, 17 (3), pp. 185-197, 2001
- [4] Marc Levoy. A hybrid ray-tracer for rendering polygon and volume data. IEEE Computer Graphics and Applications, 10 (2):33-40, March 1990.
- [5] David Laur and Pat Hanrahan. Hierarchical Splatting: A progressive refinement algorithm for volume rendering. Computer Graphics (ACM Siggraph Proceedings), 25 (4):285-288, July 1991.
- [6] W.Lorensen and H.Cline. Marching cubes a high resolution 3D surface construction algorithm, ACM Computer Graphics (Proceedings of SIGGRAPH '87), vol.21, n 4, pp 163-170, 1987.
- [7] Donald J. Meagher. Geometric Modeling using octree encoding. Computer Graphics and Image Processing, 19:129-147, 1982.
- [8] C.Montani, R.Scateni and R.Scopigno. Discretized Marching Cubes, in Visualization '94 Proceedings, R.D. Bergeron and A.E.Kaufman, Eds. 1994, pp.281-287, IEEE Computer Society Press.
- [9] C.Montani, R.Scateni and R.Scopigno. Decreasing Isosurface Complexity via Discrete Fitting, Computer Aided Geometric Design, 17 (2000) 207-232.
- [10] I.Navazo. Extended Octree Representation of General Solids with Plane Faces: Model Structure and Algorithms, Computer and Graphics, vol 13, 1, 1989, pp.5-16.
- [11] M.Ohlberger and M. Rumpf. Hierarchical and Adaptive Visualization on Nested Grids. Computing , 59(4), pages 269-285, 1997
- [12] H.Samet. Applications of Spatial Data Structures. Addison Wesley, Reading, MA, 1990.
- [13] R.Shekhar, E.Fayyad, R.Yagel and J.Cornhill. Octree based Decimation of Marching Cubes surfaces. Visualization 96, 335-342, 1996.
- [14] D.Tost, A.Puig and I.Navazo. Visualization of mixed scenes based on volume and surface. In Proceedings of the Fourth Eurographics Workshop on Rendering, pp.281-294, 1993.
- [15] R.Westemann, L.Kobbalt and T.Erl. Real-time exploration of Regular Volume Data by Adaptive Reconstruction of Isosurfaces. The Visual Computer, 15, pp.100-111, 1999.
- [16] J. Wilhems and A. Van Gelder. Octrees for Faster Isosurface generation. ACM Transactions on Graphics, 11(3). 201-227, July 1992.
- [17] J.Wilhems and A. Van Gelder. Multi-dimensional Trees for Controlled Volume Rendering and Compression. In proceedings of 1994 Symposium on Volume Visualization, pp 27-34. ACM Press, October 17-18 1994.