A 3D TEXTURE-BASED OCTREE VOLUME VISUALIZATION ALGORITHM

Imma Boada⋆

Isabel Navazo♢

Roberto Scopigno‡

⋆ Institut d'Informàtica i Aplicacions,Universitat de Girona, Spain
♢ Departament LSI,Universitat Politècnica de Catalunya, Spain
‡ CNUCE - National Research Council (C.N.R.), Italy
imma@ima.udg.es,
isabel@lsi.upc.es,
r.scopigno@cnuce.cnr.it

ABSTRACT

Although 3D texture based volume rendering guarantees to obtain image quality almost interactively, it is difficult to maintain this interactivity when the technique has to be exploited on large data sets. In this paper, we propose a new texture memory management policy that substitutes the classical assignation policy of one texel per voxel, applied for the volume representation in texture space, for a more synthetical one that benefits of nearly homogeneous regions and areas of no interest of the volume. Based on this new policy a 3D texture based Octree Volume Visualization algorithm, that combines 3D texture hardware and hierarchical representation of volume data, is presented. The algorithm allows multiresolution renderings of very large data sets and guarantees high image quality at regions of special interest. The simplified representation applied to the non-interesting regions of the volume improves rendering speed.

Keywords: 3D Textures,Volume Visualization, Octrees

1 INTRODUCTION

During the last years a lot of efforts have been undertaken to obtain real-time exploration of volume data sets. 3D texture mapping hardware, in particular, has become the fastest volume rendering method available on high-end workstations. Although the adoption of this hardware-enhanced approach guarantees to obtain image quality almost interactively, it is difficult to maintain this interactivity when the technique has to be exploited on large data sets.

Fundamentally, the 3D texture mapping hardware obtains the volume rendering from the back to front composition of a set of planes that slices and samples the volumetric data loaded into the texture memory. This set of planes is drawn as a set of 3D textured polygons that are blended together to obtain the final image. Texture mapping and compositing operations are performed by hardware very quickly. As long as the volume completely fits into the texture memory, the time consumed for the generation of an image is negligible compared to software based approaches. Nevertheless, rendering speed increases when the entirely volume can not be represented into the available texture space. In this situation, the volume has to be broken down into several bricks, which are treated independently. The entirely volume rendering is obtained by the composition of all these brick contributions [Grzes98].

Some techniques have been proposed to deal with this situation. [Srini97], [Tong99] for example, propose a better management of texture memory space based on the no representation of empty regions. However, despite the introduced improvements, interactivity is still difficult to maintain.

As the main drawback of the 3D texture based volume visualization technique is related to its memory capacity, which limits the volume that could be represented in the texture space, we propose a new texture management policy that benefits of nearly homogeneous regions and areas of no interest of the volume, and modifies the classical assignation policy of one texel per voxel for a more synthetical one. Based on this new assignation rule, we present a 3D texture based Octree Volume Visualization algorithm, that combines 3D texture hardware and hierarchical representation of volume data, to obtain multiresolution renderings of very large data sets, improving rendering speed. The algorithm exploits advantages of both, octree representation and 3D texture hardware.

The paper is structured as follows. In Section 2 a brief description of background and related work in 3D texture volume rendering is given. After a description of the octree volume representation in Section 3, we introduce in Section 4 the proposed 3D texture based volume visualization algorithm. In Section 5 some results are given. Finally in section 6 conclusions and future work are described.

2 BACKGROUND

2.1 Technique overview

The 3D texture based approach is a direct data visualization technique that takes advantage of spatial coherence, being much faster than ray casting and obtaining *almost* the same quality images. Unlike ray casting, where each image pixel is built up ray by ray, the 3D texture based approach processes a set of rays simultaneously, using the 3D texture as a voxel cache to store intermediate results.

This volumetric rendering technique can be decomposed in two steps: the *Texture Definition* and the *Volume Rendering*. In the first step, a 3D texture is obtained from the application of a transfer function and/or a look-up-table that maps voxel field values in the data volume to the $RGB\alpha$ values of the texture (texels). The texture represents an $RGB\alpha$-encoded view of the volume model[1]. In the second step, the Volume Rendering is obtained from the composition of np polygons that slices and samples the texture space. These polygons are blended to obtain the desired transparency effect and composited into the frame buffer in a back-to-front order.

The 3D texture approach is simple and yields high quality results as long as the volume data set can be entirely represented in texture memory. But, in situations where the volume is larger than the available texture space, the volume has to be broken up into a set of bricks [Grzes98]. Each brick is processed independently and the entire volume visualization is obtained from the contributions of those bricks. The technique become more complex and frame rate is reduced. This has become the main drawback of the method.

2.2 Related Work

The use of 3D texture-mapping hardware for direct volume rendering was first mentioned by Akeley for the SGI Reality Engine [Akele93]. Cullip and Neumann [Culli93] sketch two approaches and apply them to CT data obtaining the first pictures based on this technique. Guan and Lipes [Guan94] discuss hardware issues. Cabral, Cam and Foran [Cabra94] describe how to use texture mapping hardware to accelerate numerical Radom transforms. Wilson et al [Wilso94] present an easy to implement method for direct volume rendering that uses 3D texture maps. This paper describes how most of the programming of previous methods can be eliminated by the use of graphics library procedures to perform texture space transformations. In [Gelde96] a shading model is incorporated to this method.

[Srini97] and [Tong99] present some improvements to deal with large data sets. In [Srini97] an octree is proposed to skip empty regions of the volume. In [Tong99] a preprocessing step is presented to select the volume data that contains object voxels to ensure that only these regions will be loaded in texture memory. In these methods the final texture representation of the volume assign one texel per voxel, thus for large data sets texture memory limitation is still a problem.

3 THE OCTREE REPRESENTATION

An octree is a multi-resolution data representation structure used to reorganize volume data for easy identification of *interesting* regions. Octrees have been used previously for volume rendering, e.g. [Levoy90] [Laur91][Wilhe92], often with the objective to avoid the evaluation of non-interesting regions and to optimize rendering speed.

[1] Density values and gradients can be also represented in texture space [Weste98]

The octree construction always requires criteria to determine when a region of the volume is important or not. These criteria vary from the simplest that evaluates the presence or absence of data, to the more sophisticated in which the variation of internal sample values with respect to the samples of the vertices is evaluated to detect homogeneous zones. Once a criterion has been defined, leaf nodes are identified as those with maximal degree of satisfaction of the evaluated criteria. This degree of satisfaction represents the *accuracy* of the node.

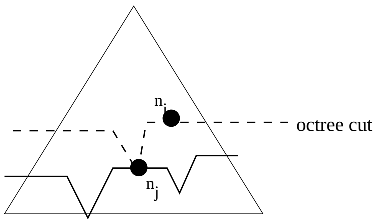


Figure 1: The triangle represents the pyramid that indicates a complete resolution set. The solid line represents leaves of the octree. The octree is a subset of the pyramid that completely spans the volume.

3.1 Notation

Given an octree representation of a volume data set, we define a *cut of the octree*, represented as $C = \{n_0, n_1, ..., n_l\}$, as a set of nodes such that, for each possible root-leaf path, one and only one of its nodes is contained in the cut. No redundant data is stored in the cut, and all the voxels are represented with more or less accuracy by one of its nodes, giving a multi-resolution representation. For each node $n_i \in C$ its octree level, denoted as l, and the level of its deepest descendent node, denoted as d, are known.

In this paper, we consider that the octree representation of the volume dataset has been obtained in a pre-processing step, in which homogeneous zones have been identified, and the accuracy of each one of its nodes has been computed. The cut, at which the visualization algorithm has to be applied, is also known. Although we do not consider how this cut has been selected, in [Boada99] we present an heuristic algorithm to select an optimal cut, optimal in the sense that the entirely cut can be represented in the available texture memory. The user identifies the regions of maximal interest and the algorithm provides the nodes of the cut and the accuracy that must be used for their representation in texture space in order to optimize rendering speed.

4 THE 3D-TEXTURE BASED OCTREE VISUALIZATION

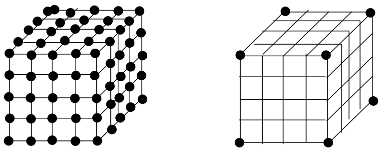
The visualization algorithm can be applied for the rendering of any cut of the octree. In particular, we consider that it is applied to the C cut, where $C = \{n_0, ..., n_l\}$.

The rendering process is decomposed in two steps: the representation of the cut in texture memory and the cut rendering. Next subsections gives all the details of these steps.

4.1 Octree Cut Representation in Texture Space

The octree cut representation in texture space is obtained from the representation of every node in an independent texture. Each node $n_i \in C$ will be in one of these two situations:

- *The node is a leaf* (see node n_j in figure 1). In this case, all relevant information of the subvolume represented by this node can be obtained from the samples distributed over the vertices of the node, we call this samples *external samples* (see figure 2). Therefore, to represent the node in texture space only these external samples are used.



(a) (b)

Figure 2: (a) The node is represented at voxel level, the assignation one texel per voxel is applied. (b) A subsampled representation of the node is used. In this case the required texture space is reduced.

- *The node is not a leaf* (see node n_i in figure 1). To represent the volume information of the node in texture space several options are possible. These options vary from the finest representation, that is a grid $2^{(l-d)^3}$, to the coarser, obtained from the children of the node (see figure 3).

  The selected representation determines the samples that will be considered for the representation of the node in the texture space. To select which is the best representation a user fixed criterion based on the node's degree of interest or node's degree of homogeneity can be applied. The criterion determines at which accuracy has to be represented the node to satisfy user restrictions[2].

If the selected level for the representation is q the region covered by n_i is represented by the set of all external samples of descendent nodes of n_i that are at level q.

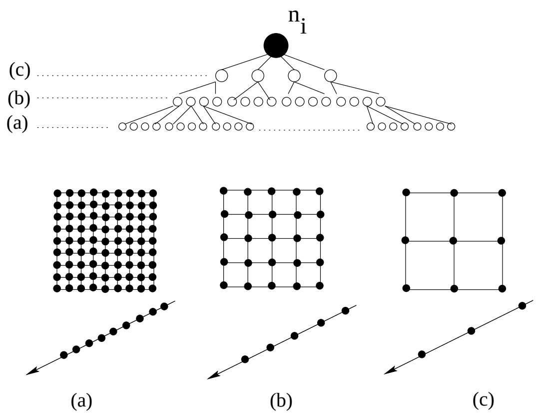


Figure 3: Given a node n_i of one of the possible cuts on the octree of figure 1, we show the corresponding subtree; because n_i is not a leaf node, different texture representations can be chosen according to the depth of the subtree rooted in n_i

It could be concluded that the assignation of one texel per voxel, will be applied only when the node of the cut is represented by minimal division nodes (i.e. voxels) (see figure 3(a)). In all other situations, the assignation policy will assign a more synthetical data size to texture size correspondence. Based on this new assignation rule, the representation of the cut in texture memory can be selected in such a way that the entirely cut could be represented in texture memory [Boada99].

4.1.1 From Samples to Texels: Opacity Correction Factor

Once the set of samples that will represent the node's volume are selected, we have to represent them in texture memory. Differently of classical methods, in which the application of a transfer function and/or a look-up-table maps voxel field values to the RGBα texel values, when a subsampled representation of the node is stored an opacity correction factor is applied to preserve image quality. The computation of this opacity correction factor is done as follows.

[2] Usually in terms of rendering speed and image quality

From [Levoy90] the accumulated opacity α_{out} of a volume element can be expressed as

$$\alpha_{out} = \alpha + \alpha_{in}(1 - \alpha) \qquad (1)$$

where α is the opacity of the volume element and α_{in} is the entering opacity to this element. The α_{out} value depends directly on the number of samples that are considered. Obviously, the best representation will be obtained when the node is represented by minimal division nodes (i.e. the classical assignation of one texel per voxel). In figure 3, three representations of a node that differ in the number of samples considered for its representation in texture space are shown. The 3 (a) representation is the best, as it is defined at voxel level, 3(b) and 3(c) correspond to a subsampled representation of the same node. Of course, it should be desirable that renderings obtained from 3(b) or 3(c) to be very similar to the one we obtain using the higher resolution representation. With this purpose, i.e. reduce the difference between renderings, an opacity correction factor (op_c) is applied to subsampled representations.

To describe how the op_c value is obtained, we analyze the two situations of figure 4 in which, for simplicity, only the first samples of a ray from 3(a) and another from 3(b) representations are evaluated. If α_0, α_1 and α_2 are the opacities of s_0, s_1 and s_2 samples of ray 4(a), and α_0^* and α_2^* are the opacities of s_0^* and s_2^* samples of ray 4(b), from equation Eq.(1), we obtain that the accumulated opacity by the back-to-front composition of these samples is

$$\alpha_a = \alpha_2 + \alpha_1(1 - \alpha_2) + \alpha_0(1 - \alpha_1)(1 - \alpha_2) \quad (2)$$

for 4(a), and

$$\alpha_b = \alpha_2^* + \alpha_0^*(1 - \alpha_2^*) \qquad (3)$$

for 4(b), where $\alpha_0^* = \alpha_0$ and $\alpha_2^* = \alpha_2$.

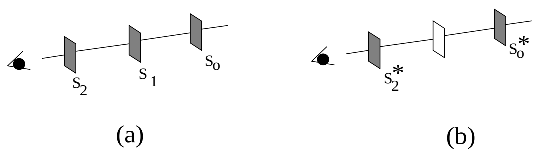


Figure 4: (a) The first samples of a ray from the finnest representation of a node; (b) the first samples of a coarser representation

It can be seen that the accumulated opacities only will be the same if $\alpha_0^* = \alpha_0 + \alpha_1(1 - \alpha_0)$. Thus, the optimal op_c is $op_c = \alpha_1(1 - \alpha_0)$ where $\alpha_0 = \alpha_0^*$

and α_1 is approximated according the criterion applied for the octree construction: if the octree construction criterion considers a node *homogeneous when all the samples are equal*, $\alpha_1 = \alpha_0 + \epsilon'$, where ϵ' depends on the accuracy of the node; if the octree construction criterion considers a node *homogeneous when internal samples can be obtained by interpolation of the external ones*, α_1 can be approximated by $\frac{\alpha_2 - \alpha_0}{2} + \epsilon'$, ... Obviously, the complexity of the computation of this op_c value depends on the criterion applied for the octree construction and how α_1 is approximated. Nevertheless, as a coarser representation is applied only at regions of no maximal interest, it has been experienced, that is enough to consider $\alpha_1 = \alpha_0 + \epsilon'$. Although this approximation introduces some error, this error only affects non-interesting regions.

In general given a node $n_i \in C$, that represents 2^{l^3} samples of the original volume data set, and given q the selected level for its representation in texture space, where $d \leq q \leq l$, the α_i^* opacity values, where $i = 0, .., 2^{(l-q)^3}$, that correspond to this subsampled representation are substituted for $\alpha_i^* + op_c$ where

$$op_c = (1 - \alpha_0)((1 - \alpha_0)^{q-1} - 1)$$

4.1.2 Texture Restrictions

Two restrictions have to be considered when the texture is defined. The first one, is imposed by hardware and forces texture data loaded in texture memory to be packed into a rectangular parallelepiped, each dimension constrained to be a power of two. The second one, has to be accomplished to preserve image quality and forces each texture to share its boundary with its neighboring texture nodes.

4.2 Octree Cut Rendering

The octree cut rendering, which only considers orthographic projections, is based on an iterative application of a node visualization function which computes the contribution of each node to the final image. The composition order of the nodes is obtained directly from a back-to-front octree traversal, driven by the viewing direction.

4.2.1 Node Visualization Function

The function is based on the [Gelde96] algorithm; all texture space transformations and clipping plane performance are implemented using graphics library procedures

To describe the function we consider that it is applied to a node n_i of the cut. The texture representation of this node is T_i, and np_i is the number of polygons that will be used for the node's rendering[3]

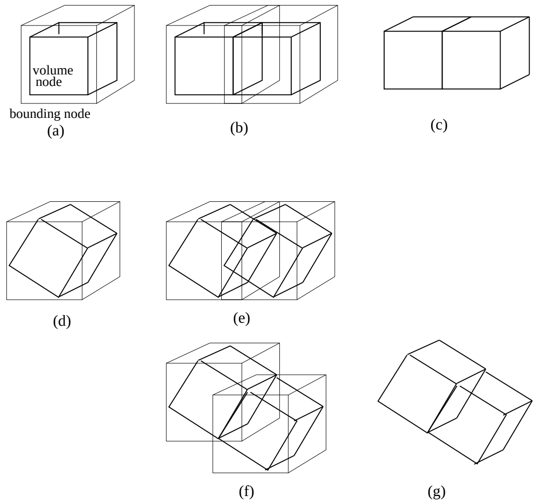


Figure 5: This illustrates an orthographic projection of two 3D bounding-node, the first in the original orientation (a) and the second one rotated (d). Neighbor node is correct positioned by a set of transformations, (b)(e) and (f). (c)(g) corresponds to the final image contribution of the node

The function can be decomposed in the following steps:

1. BOUNDING NODE DEFINITION. A bounding cube centered on the center of the node n_i is defined. We call it bn_i, *bounding node*. The corresponding polygons np_i to be rendered form series of slices through bn_i, parallel to the xy face.

2. CLIPPING PLANES ACTIVATION. The set of np_i polygons to be rendered always remains parallel to the projection plane, and extend to the bounding node. The node's volume is represented in T_i and rotates into bn_i according to the viewing direction (see figure 5(a)(d)) . To guarantee that only the volume information represented in the texture

[3] np is computed by finding the longest ray through the node (for any viewing direction), transforming the voxel values found along that ray to texels.

will contribute to the final image (i.e. external areas of the node contained in bn_i are eliminated) a set of clipping planes is defined. The clipping planes are positioned according the viewing direction at each one of the node's volume faces.

3. NODE'S POSITIONING. Once the *clipped polygons* that represent the volume node are computed, we determine where they have to be projected. The position of these polygons is fixed by the node's octree position and by the viewing direction. These parameters determine the set of geometrical transformations to be done.

   A first geometrical transformation determines the position of the bn_i according to the octree position (see figure 5(b)(e)). A new orientation requires a rotation of the texture space, keeping bn_i stationary. This rotation is done independently for each node according to the viewing direction. To maintain continuity between nodes, a second transformation is applied (see figure 5(f)), this transformation translates the bounding node.

4. TEXTURE MAPPING. Finally, texture coordinates are assigned to the *transformed and clipped polygons* in such a way that they interpolate into the texture-space when they are within the node. All mathematics to generate texture coordinates are based on [Gelde96].

Once these steps are applied we obtain the final I_i contribution to the image (see figure 5(c)(g)).

5 RESULTS

All tests have performed on a SGI Octane MXE with 4 Mbyte of texture memory. As the maximal volume data set that entirely fits in this texture memory is of $128x128x64$ we have chosen two data sets that exceed it. The first one corresponds to a CT head of $128 * 128 * 128$ and the second one to a jaw data set of $256x256x128$.

For first renderings a cut of the CT head octree has been selected. The cut is composed for 6 nodes of 64^3 voxels and the rest of 32^3. Only the selected region of figure 7 is considered of maximal interest. In the first rendering, see figure 8, all the nodes have been represented at the higher resolution, i.e. one texel per voxel. We consider that the higher image quality is obtained when

Model	Time	Id	T:V
Fig. 8	1.32	(1)	1:1
Fig. 9	0.96	(2)	1:2
Fig. 10	0.62	(3)	1:4
Fig. 11	0.81	(4)	1:2
Fig. 12	1.98	(5)	1:2 and 1:4

Table 1: (Time) Rendering time in seconds, (Id) Identifier of the required texture space represented in fig.6, (T:V) Texel-voxel correspondence applied for the representation of non-interesting nodes.

each voxel is represented by one texel, thus maximal quality is reached in this rendering. Precision errors that could appear in node's positioning (see 4.2.1 (3)) are not perceptible.

Next renderings, figure 9 and figure 10, correspond to the same cut and differ in the degree of simplification applied for the representation of non-interesting nodes. For maximal interest nodes the assignation policy used for the representation has been of one texel per voxel. In figure 9 the texel-voxel assignation for the non-interesting nodes representation is 1:2 (i.e. a node of n^3 is represented with $(n/2)^3$ texels). The number of polygons used for the rendering of these nodes is $\sqrt{3}(n/2)$. In figure 10 the texel-voxel relation is 1:4 and the number of polygons is $\sqrt{3}(n/4)$. It is perceptible that the image quality of regions of no interest is not comparable to the rendering that we obtain when these regions are rendered at higher resolution (see figure 8). The benefits of this new representation in terms of rendering speed and required texture space are summarized in Table 1 and in Figure 6 respectively.

The images generated from the jaw data set octree are shown in figures 11, 12. In Figure 11 the entirely volume has been forced to fit into the available texture space, the applied texel:voxel relation has been 1:2. This rendering facilitates the identification of regions of maximal interest that will be used for the cut selection.

In the final rendering, figure 12, the 1:1 texel-voxel assignation is applied to the maximal interest nodes which correspond to a region of 64^3 voxel. The rest of the nodes are represented at 1:2 or 1:4 according the distance from the region of interest.

We can conclude that the new assignation policy applied for the texture representation of the vol-

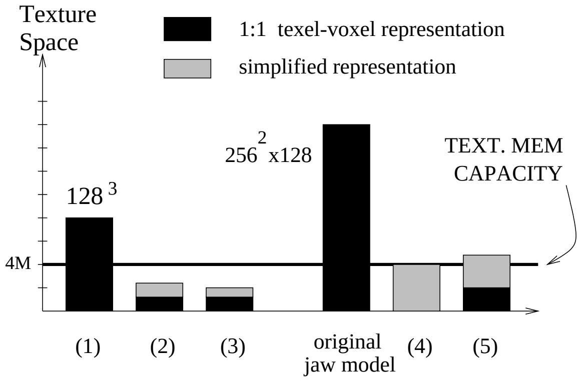


Figure 6: Required texture space for each cut rendering.

ume reduces the required texture space and thus improves the rendering speed. Obviously, these improvements depends on the size of the initial volume data set and on the area of interest defined on it.

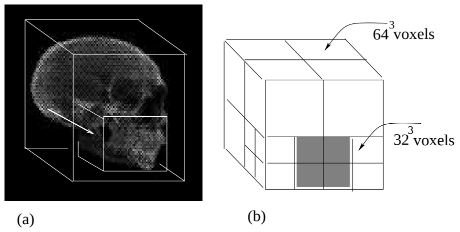


Figure 7: (a) The region of maximal interest is selected. (b) The cut of the octree is composed by 6 nodes of 64^3 and the rest of 32^3.

6 CONCLUSIONS AND FUTURE WORK

We have presented a 3D texture based Octree Volume Visualization algorithm that combines 3D texture hardware and hierarchical representation of data sets to obtain multiresolution renderings. The basis of the algorithm is a new rule that takes advantage of iso-valued areas of the volume or regions of no interest, and modifies the classical texture volume representation for a more synthetical one. This new assignation policy improves the use of 3D texture memory and optimizes the exploitation of 3D texture based visualization on large data sets.

Given an octree representation of the original data set, one cut of this octree is selected. Each node of the cut is represented in texture memory at one determined accuracy according the degree of homogeneity and the degree of interest of the node. Once all the nodes of the cut are repre-

Figure 8: Texel-voxel assignation 1:1 for all the nodes of the CT head octree cut

Figure 9: Texel-voxel assignation 1:2 for nodes of no interest

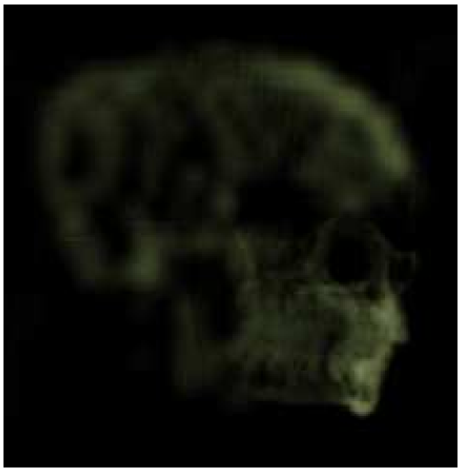


Figure 10: Texel-voxel assignation 1:4 for nodes of no interest

sented in texture space a rendering function is applied to each one to obtain the final image. The algorithm guarantees at maximal interest regions the image quality offered by classical 3D texture based visualization algorithm.

In our future work different policies to select the cut of the octree will be analyzed [Boada99]. We will also work on the extension of the algorithm for perspective projections.

Figure 11: Texel-voxel assignation 1:2 to fit the jaw model into texture space

Figure 12: Nodes of no interest represented at 1:2 or 1:4 according to the distance of the region of maximal interest

REFERENCES

[Akele93] Kurt Akeley, Reality Engine Graphics, Computer Graphics (ACM Siggraph Proceedings), 27:109-116,1993.

[Boada99] I.Boada, I.Navazo, R.Scopigno, *Multiresolution Texture Volume Representation.* Technical Report IIiA99-16-RR. University of Girona.

[Cabra94] Brian Cabral, Nancy Cam and Jim Foran, *Accelerated Volume Rendering and Tomographic Reconstruction using Texture Mapping Hardware*, In ACM Symposium on Volume Visualization,pp. 91- 98 Washington, D.C. October 1994

[Culli93] T.J.Cullip, U. Neumman. *Accelerating Volume Reconstruction with 3D texture hardware.* Technical Report TR93-027, University of North Carolina, Chapell Hill, 1993.

[Gelde96] Allen Van Gelder, Kwansik Kim. *Direct Volume rendering with shading via 3D textures.* Symposium on Volume Visualization 1996, 23-30.

[Guan94] Sheng-Yih Guan and Richard Lipes. *Innovative Volume rendering using 3D Texture Mapping.* In Image Capture, Formatting and Display. SPIE 2164, 1994

[Grzes98] R.Grzeszczuk, C.Henn, and R.Yagel. *Advanced Geometric Techniques for Ray Casting Volumes.* Course Notes. SIGGRAPH '98. ACM July 1998.

[Levoy90] Marc Levoy. *A hybrid ray-tracer for rendering polygon and volume data.* IEEE Computer Graphics and Applications,10 (2):33-40, March 1990.

[Laur91] 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.

[Srini97] Rajagopalan Srinivasan, Shiaofen Fang, Su Huang. *Volume Rendering by Template-Based Octree Projection.* Workshop Eurographics 97. Visualization in Scientific Computing.

[Tong99] Xin Tong, Wenping Wang, Waiwan Tsang, Zesheng Tang. *Efficiently Rendering Large Volume Data Using Texture Mapping Hardware.* Data Visualization '99.

[Weste98] R.Westermann and T.Ertl. *Efficiently Using Graphics Hardware in Volume Rendering Applications.* In Computer Graphics. Proceedings of SIGGRAPH 98, 169-177, August 1998.

[Wilhe92] Jane Wilhems, Allen Van Gelder. *Octrees for Faster Isosurface generation.* ACM Transactions on Graphics, 11(3):201-297, July 1992.

[Wilso94] Orion Wilson, Allen Van Gelder, Jane Wilhems.*Direct Volume rendering via 3D textures.* Technical Report UCSC-CRL-94-19, University of California, Santa Cruz, June 1994