.1400014599
COPIA 1

Constructing face octrees from voxel-based volume representations

R. Juan i Ariño
J. Solé i Bosquet

Report LSI-93-17-R

Constructing Face Octrees from voxel-based volume representations

Robert Juan i Ariño

Jaume Solé i Bosquet
Departament de Llenguatges i Sistemes Informàtics
Universitat Politècnica de Catalunya
Barcelona, Catalonia

June 1993

Abstract

Boolean operations between solids can be efficiently performed by operating their respective octree encodings. Among the variety of octree-based representation models, face octrees are approximate representations that give a good compromise between the need of storage saving and the simplicity of the involved algorithms. They are also a valuable model for representing smooth free-form surfaces. Constructing face otrees from voxel-based volumes representations will be useful in order to obtain a more compact, smoother and further operable encoding.

A method to perform such a conversion is presented. The method consists of two steps. First, a network of points that represents the volume data is extracted. The extraction is based on the Geometrically Deformed Models technique. Then the network of points is transformed into a face octree. In the transformation process, face octree nodes are compacted as much as possible while preserving the volume data precision.

Contents

1 Introduction

Voxel-based volume models are widely used in scientific visualization, [11], [16]. This is mainly due to the fact that voxels are the most simple way of representing spatial data stored in 3D arrays as well as because this kind of data is what remote sensing and scanning technology generates from nondestructive examination of the internal estructure of objects. Unfortunately, voxel-based volume schemes have some disadvantages. The most important is the lack of concission leading to huge storage requirements to encode even simple solids. Moreover, rendering algorithms are time-consuming specially when realistic images should be generated.

Hierarchical spatial encodings such as classical octrees [17], [27], [26] and extended octrees [2], [5], [8] have been proposed to reduce storage requeriments, providing as well simple algorithms to perform boolean operations between solids. Face octrees are an intermediate scheme between classical octrees and extended octrees, they are more concise than classical octrees and are particularly well suited to approximate representations of objects with complex surface boundaries [3], [4].

Devising an efficient algorithm to convert huge voxel data representations into face octrees is of interest because of several reasons. First, the conversion will allow to reduce the amount of storage needed for the representation. Second, rendering of solids represented as face octrees will be faster than rendering classical octrees as well as the quality of the generated images will be improved. Third, to perform boolean operations between solids represented as face octrees still is a very easy issue.

In this paper a algorithm to perform the conversion of a voxel-based volume representation into a face octree scheme is presented. The voxel-based representation consist of a binary three-dimensional univaluated array. The array encodes in each cell either the presence or the absence of solid.

The algorithm consists of two steps. First, a network of points that represents the volume data is extracted. The extraction is based on the Geometrically Deformed Models technique, [18]. The model is a network of points each of them representing a boundary voxel. A cost function is associated with every point of the network that gives a cost to local deformations. The model evolves by minimizing the function cost while restricting each point to stay inside the limits of the represented voxel. Then the deformed network ofpoints is transformed into a face octree. In the transformation process, face octree nodes are compacted as much as possible while preserving the volume data precision given by the voxel edge length. The compactation is performed by trying to fit a plane that both is interior to all the boundary voxels belonging to the current face octree octant and does not intersect any other voxel in it.

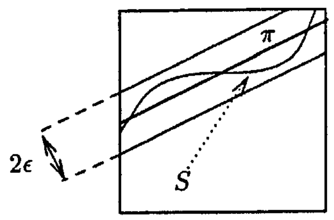


Figure 1: A face node and the associated band

2 Face Octrees

Octrees are one of the classical approximate decomposition schemes [24], [17]. They are trees that represent solids by encoding the recursive subdivision of a cubic finit universe where the solids are placed. The nodes allowed in classical octrees are either terminal or non-terminal nodes. Terminal nodes are labeled as black or white depending on whether they are fully inside or fully outside the solid boundary. The solid boundary is approximated by a layer of cubes of a minimum specified size; they are also labeled as black or white depending on some criterion [17], [27]. Non-terminal nodes are labeled as grey nodes.

In order to avoid the verbosity and the approximate character of the classical octree scheme, several proposals of new octree encodings have appeared in the literature. Polytrees where presentred in [5] and extended octrees in [2]. Besides the white and black nodes, these octrees also represent face, edge and vertex nodes. These new types of nodes are terminal nodes and allow to represent exactly the solid boundary of polyhedra while reducing verbosity and keeping a reasonable complexity for the boolean operations.

A face octree [3], [4] is an octree with white, black, face and grey nodes together with a tolerance ε that controls the degree of approximation of the representation. White, black and grey nodes are defined as in classical octrees. Face nodes contain a connected part of the object boundary and each of them has associated the equation of a plane, π, that approximates the boundary within the node with a given tolerance ε. See Figure 1. Grey nodes are those that can not be labeled as white, black or face nodes. They represent regions of the object surface that are not flat enough. When the recursive subdivision reaches the minimum predefined node size, grey nodes are terminal grey nodes. Face octrees are halfway between classical and extended octrees; they are more concise than classical octrees and are well suited to approximate representations of objects with complex surface boundaries [21], [22], [23].

3 Geometrically Deformed Models

Basically, there are three methods of displaying and analyzing three-dimensional scalar field data generated by nondestructive examination methods of solids [18]. One method processes the volume data in its original form [28], [29], [7]. The second transforms the initial data into something that is more compact, such as a surface, and therefore is more readily handled [1], [13], [9], [14], [6]. The third method generates geometric models that approximate the original solid [12], [30], [19]. This is the most powerful approach because all the knowledge from solid modelling applies to the recovered model.

As our work is mainly related with solid modelling, we will focus in the methods belonging to the third type. Among them, Geometrically Deformed Model (GDM) is the one that best fits our needs. GDM's allow to extract a topologically closed geometric model from a volume data set [18]. The technique starts with a simple polyhedron that is a coarse approximation of the object. A cost function is associated with every vertex of the polyhedron and a set of constraints are set up. By minimizing the cost function while the constraints are satisfied, the polyhedron is deformed to fit the object within the volume.

4 Face Octree Construction

4.1 Definitions

We denote by V the set of voxels given in the voxel-based volume representation. It is assumed that all the voxels in V are labeled as either black or white; they represent space regions that are, respectively, inside or outside the original solid. Two voxels in V are *neighbours* if they share a face, a edge or a vertex. We will refer to either a *vertex neighbour*, a *edge neighbour* or a *face neighbour* whenever we want make explicit the neighbourhood relation between two given voxels. For each voxel v in V we define two neighbourhood sets: the *neighbourhood*, $N(v)$, which contains the subset of voxels in V that are neighbours of v and the *face neighbourhood*, $FN(v)$, that is defined as the set of face neighbours of v. A voxel such that at least has a white voxel in its neighbourhood is said to be a *boundary voxel*; BV denotes the set of all the boundary voxels in V. With each boundary voxel v_i in BV we associate a point p_i that we call the *representative point* of the boundary voxel.

4.2 The Deformed Model and the Constraints Used

The deformed model we use is not a closed polyhedral model as in [18], it rather may be thought of as an elastic network shaped by the representative points as follows. For each vertex v_i in BV, a representative point p_i is placed in the

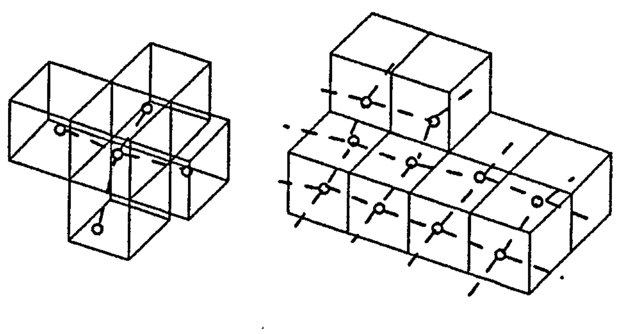


Figure 2: Interconnecting representative points.

voxel centre and a link betweeen the point p_i and each representative point of v_j in $\{BV \cap FN(v_i)\}$ is defined. See Figure 2.

A general constraint imposed on the model is that the representative points should always stay inside the limits of the associated boundary voxel. Two different cost functions, namely, topology maintenance and potencial spring energy have been studied.

4.2.1 Topology Maintenance

We would like to extract a simple and topologically coherent set of points from the volume data set. On the other hand, the nodes in the face octree model we want to extract should represent boundary regions that are flat enough. Therefore, a cost function that both maintains the local simple nature of the initial model and allows to extract smooth boundary pieces is highly desirable.

A cost function that captures these properties is the *topology maintenance constraint*, [18]. For each voxel v_i in BV, the cost is defined as the ratio of the distance from the considered model point p_i to the centroid of the points p_j in its face neigbourhood, $FN(v_i)$, and the maximum distance between the points in $FN(v_i)$.

$$M_i = \frac{||p_i - \frac{1}{n}\sum_1^n p_j||}{\max_{i,j}(||p_j - p_i||)}$$

This function constraints the local curvature of the model by directing the point p_i towards the centroid of the points in its face neighbourhood, [18]. See Figure 3.

4.2.2 Energy Minimization

The second const function we have studied is a potencial energy function. Linking each boundary voxel to the boundary voxels in its neigbourhood can be

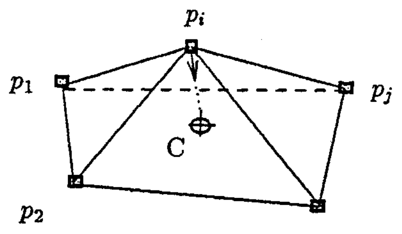


Figure 3: Topology maintenance. An estimation of the local curvature.

cast in the form of a potencial energy function that can be used to drive the evolution of the GDM, [31].

For each boundary voxel, v_i, the cost function is defined as an spring energy term $E_i = k \sum_j \|p_i - p_j\|^2$. Hence the cost function for the whole model is,

$$E = k \sum_i \sum_j \|p_i - p_j\|^2$$

As each representative point should stay inside the volume defined by the voxel, the problem of finding the final location of each representative point in the deformed model is reduced to minimizing a cost function subject to a set of linear inequality constraints, [10], [15].

4.3 The Algorithm

Several ways of constructing a face octree from voxel data can be devised. An intuitive approach led us to approximate the volume data by means of a set of planes computed as regression planes of a set of points each point being in the center of a boundary voxel. Unfortunately, although simple, this method yielded strongly unconnected, flaky-looking face octrees. Figure 4 shows the *BV* set (dotted voxels) associated with a circumference, and the face nodes generated. In some cases, unnecessary subdivision was generated because of the poor fitting of the computed planes. Figure 5 a) shows the computed regression plane that, as we shall see later, can not be accepted as a face because intersects voxels which are inside the tree node and that are not boundary data voxels. Therefore, face tree subdivision should go further. Figure 5 b) shows an alternative plane that would allow to define a face node but it can not be generated by this technique.

The problems found when using the naive method above, led us to devise an algorithm that allowed to take into account local neighbourhood considerations.

This algorithm consists of two main steps.

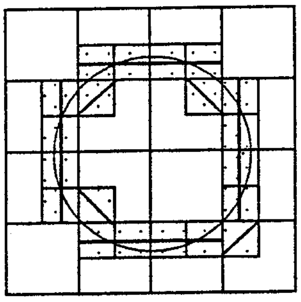


Figure 4: Regressor planes fitting. Strongly unconnected octree.

In the first step two tasks are performed. One task is extracting from the volume data a network of points that represents it. The extraction is based on the Geometrically Deformed Models technique, [18]. The other task associates a plane with each boundary voxel in the voxel-based representation. This plane is called the GDM plane and is computed as follows. First, for each boundary voxel v, the representative points in $FN(v)$ are sorted circularly with p, the representative point of v, as centre. Then for each triangle defined by p and the representative points of two consecutive vertices in $FN(v)$ a normal is computed. Finally, a normal defined as the average of the normals to the triangles weighted with the triangle areas is associated with the voxel v. The plane defined by the average normal and the representative point is the GDM plane.

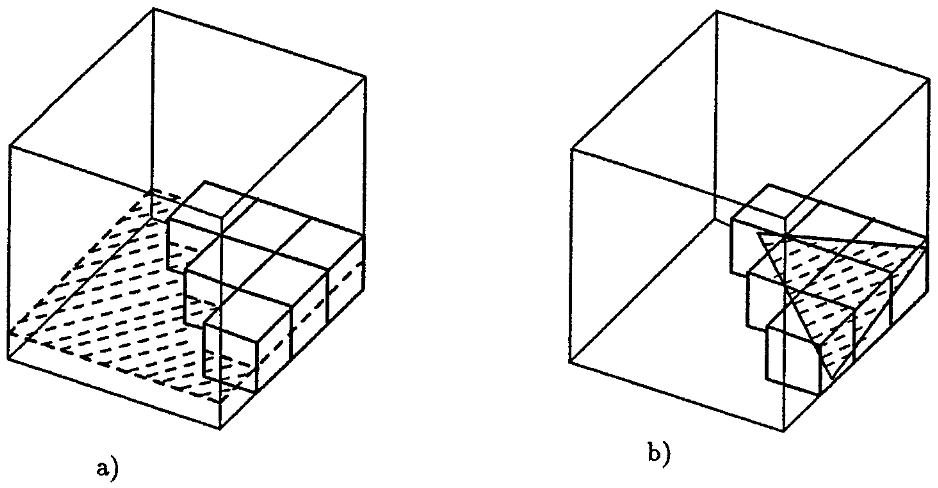


Figure 5: Regressor planes fitting. Unnecessary subdivision.

The second step transforms the deformed network of points and the associated GDM planes into a face octree. The transformation process compacts face octree nodes as much as possible while preserving the volume data precision given by the voxel edge length. The compactation is performed by trying to fit a plane that both is interior to all the boundary voxels inside the region bounded by the current face octree node and does not intersect any other voxel in the node.

Let us assume that the face octree is stored linearly by a depth method [20], [26], and that the face octree, a pointer to the next free position in the face octree and the GDM model are stored in the static variables `fo`, `po`, and `gdm` respectively. Furthermore, assume that `V` stores de initial volume-based representation. Then the algorithm can be stated as follows

```
procedure Face_octree_construction (V)
    var
        real size
        point v, origin
        plane $\pi$
        static gdm gdm
    fivar

    Compute_gdm (V, gdm)
    for each $v_i \in BV$ do
        Compute_gdm_planes ($v_i$, gdm, $\pi$)
    endfor
    Volume_data_embedding (V, origin, size)
    Generate_void_octree ()
    Tree_construction (origin, size)
endprocedure
```

Procedure `Compute_gdm (V, gdm)` extracts the GDM from the volume data and procedure `Compute_gdm_planes` (v_i, gdm, π_i) computes the planes associated with the GDM.

Procedure `Volume_data_embedding (V, origin, size)` is trivial: it computes the origin and size of a cubic universe where the voxel data is located and from which the uniform, recursive subdivision will start. The planes that defines the octree subdivision will be coincident with planes belonging to the data voxel discretization; that is, the planes that define the octree subdivision never intersect the interior of a data voxel. Let nv be the number of voxels along the volume data matrix axis. The cubic universe edge length where the voxel data will be embedded should be 2^n with

$$n = \min\{m : 2^m \geq nv\}$$

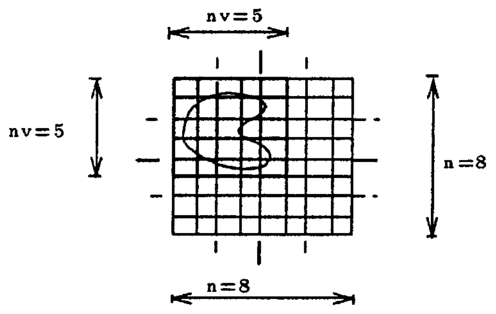


Figure 6: Voxel discretization and tree subdivision.

then

$$n = \lceil \frac{\log nv}{\log 2} \rceil$$

Figure 6 shows a 2D exemple where $nv = 5$ and $n = 3$.

Let us now concentrate in the tree construction. We will denote by OV the subset of voxels of V that are in the space region bounded by the current octree node which is defined by its origin and size. When all the data voxels in OV are black we will say that it is a *black set*. If all the data voxels are white we will say that OV is a *white set*. Finally, if some of the voxels in OV are boundary voxels, the set OV wil be called a *boundary set*.

What we want now is to properly label the face octree nodes while keeping as much as possible from going deeper in the recursion. Let OV be the set of voxels in the current octree node. When OV is either a black set or a white set, the current node can be labeled as black or white respectively. If OV is a boundary set, we would like to label the octree node as a face node. But this means to have a plane that fulfills the constraints already mentioned: the plane should intersect all the boundary voxels and only the boundary voxels inside the boundary set. In order to find out such a plane, first a candidate plane is computed as an average of the GDM planes belonging to the current octree node. Then the validity of the candidate plane is checked. If the test succeeds, the node can be labeled as face node and the computed plane is the face plane. Otherwise the set OV is split in eight subsets, each subset being defined by the corresponding octant of the current octree node. Now, each of these subsets can be recursively labeled. The algorithm can then be stated as follows

```
procedure Tree_construction (origin, size)
    if Black_set (origin, size) then
        po↑type := black
        po := po + 1
    else if White_set (origin, size) then
        po↑type := white
        po := po + 1
    else if Boundary_set (origin, size) then
        Compute_face_plane_candidate (origin, size, $\pi$)
        if Valid_candidate (origin, size, $\pi$) then
            po↑type := face
            po↑plane := $\pi$
            po := po + 1
        else
            for $i \in 0, 1, \ldots, 7$ do
                Tree_construction (origin$_i$, size/2)
            endfor
        endif
    endif
endprocedure
```

This algorithm always terminates and each boundary voxel contributes to compute some face plane in the octree. Let us first prove that the algorithm terminates. To prove that, we need to show that the recursion terminates. Assume that we are at an arbitrary level of recursion such that the edge of the current octree node is larger than the voxel edge and assume that OV is the set of voxels in the current octree node. If OV is either a black set or a white set, the octree node is labeled accordingly and the recursion terminates. If OV is a boundary set and a face plane has been found then the node is labeled as face node and the recursion terminates. If such a plane is not found, OV is split in eight subsets. Each subset contains those voxels from OV that are in the region bounded by an octant of the node. Conversely, each voxel in OV belongs to only one of the eight sets. The algorithm is then recursively applied to each of these sets. Now assume that the edge of the current tree node and the voxel edge have the same size. Obviously, if the voxel is black or white the recursion terminates. If the voxel is a boundary voxel the GDM plane is a valid face plane and the recursion terminates. Hence, the recursion always terminates.

Let us now prove that each boundary voxel contributes to compute some face plane in the octree. Assume that we are at the first call to the algorithm. All the data voxels are in OV and obviously it is a boundary set. Assume that OV is a boundary set. Then either a face plane has or has not been found. In the first case, by definition, all the boundary voxels have been involved in computing the face plane. In the second case, OV is split in eight subsets with each voxel from OV belonging to one and only one of these subsets. Then the algorithm is recursively applied to each of these subsets. As before, assume now that the edge of the current tree node and the voxel edge have the same size.

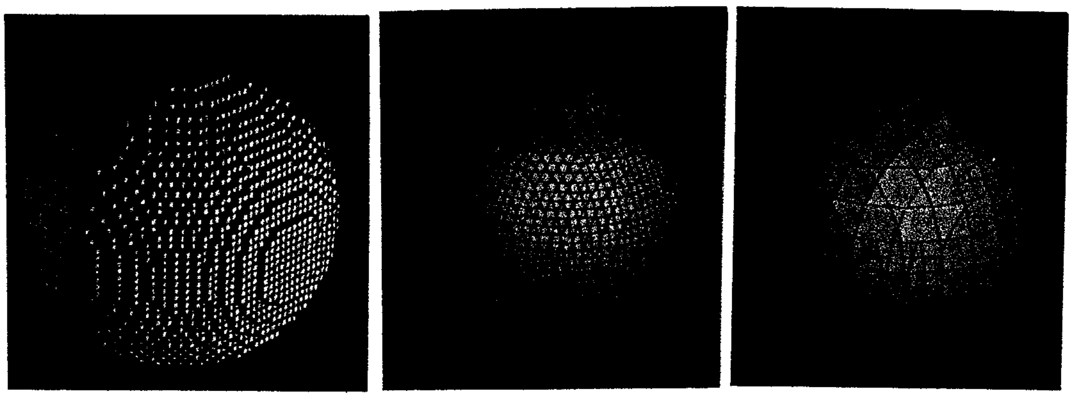


Figure 7: Case study: Sphere.

Obviously, if the voxel is black or white it does not contribute to any face plane. If the voxel is a boundary voxel the GDM plane is the face plane. Hence, all the boundary voxels contributes to some of the face planes in the face octree.

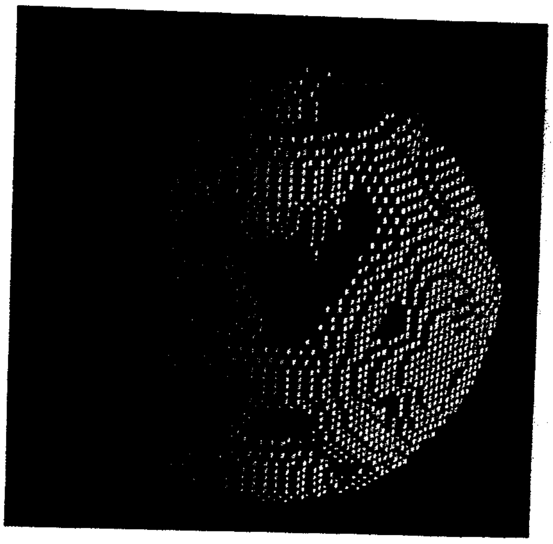
5 Results

The ideas presented in this article have been implemented on a HP9000-835 workstation. Figures 7 to 9 show three examples of face octree construction from voxel data using the proposed algorithm. The original solids are respectively a sphere, a set of tori, and a CSG-generated complex object. In the three figures, a) depicts the set of data voxels, b) depicts the GDM planes, and c) represents the final face octree. The cubic universe edge length where the voxel data has been embedded has been 2^6 in all the cases. Minimization process has been carried out with a standard optimization routine from the NAG library, [25].

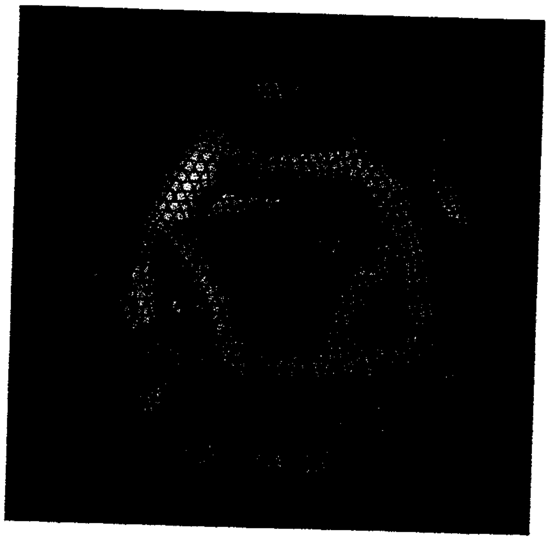
The results obtained with the topology maintenance constraint are summarized in Table 1. The second column gives the number of voxels in the volume data representation; the third column gives the number of nodes in the face octree constructed; the fourth gives the ratio face-octree-nodes/voxels, and the last column gives the cpu time process.

Compression rates, though being reasonably good, would increase when dealing with more precise input data, it is, when the solids are embedded in the 2^8 usual cubic universe.

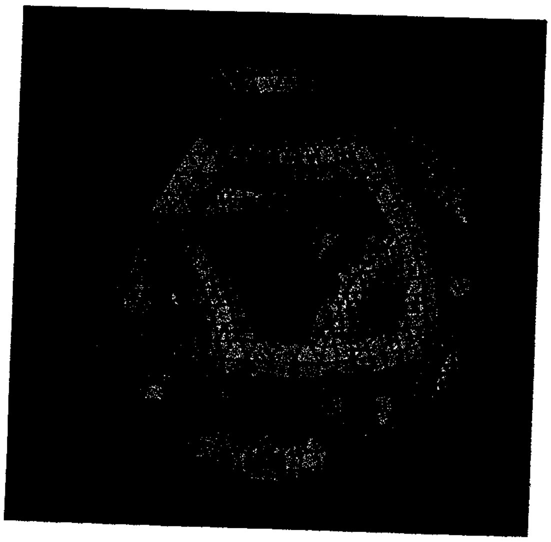
The average difference between the locations of the representative point of a voxel data computed by topology maintenance and by spring potencial energy is of about 3% of the data voxel diagonal length. The standard deviation is 0.2%.



a)



b)

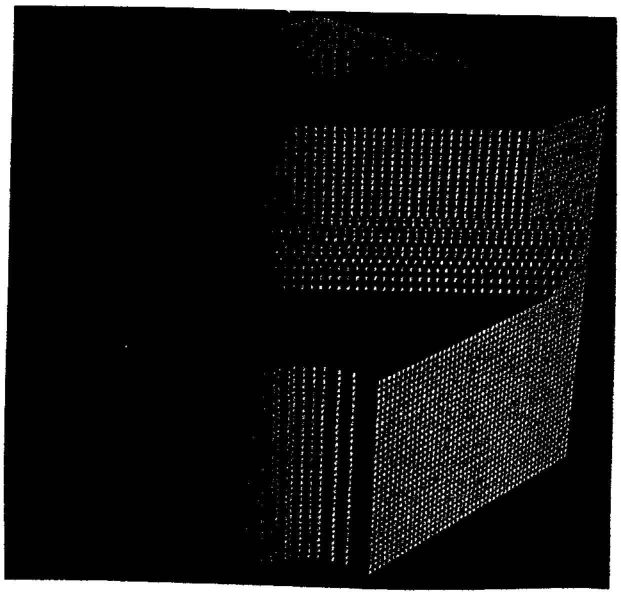


c)

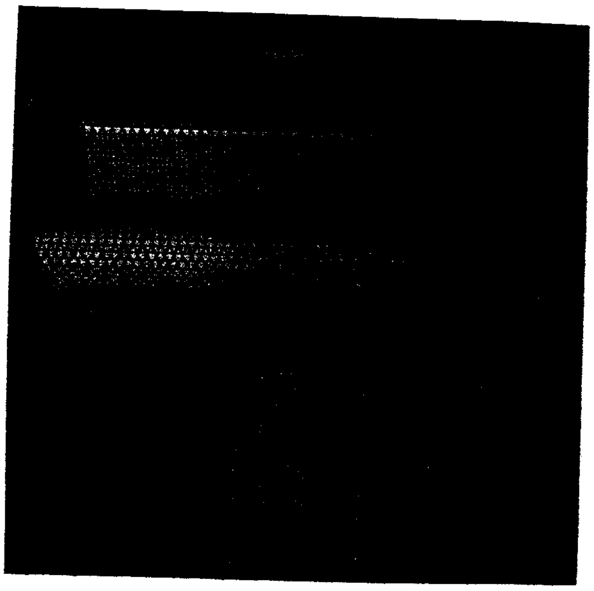
Figure 8: Case study: Tori.

Case	# voxels	# fo nodes	ratio (%)	cpu time (sec)
Sphere	37455	2513	6.71	17.83
Tori	39793	18929	47.56	42.81
Complex solid	179373	6937	3.87	67.94

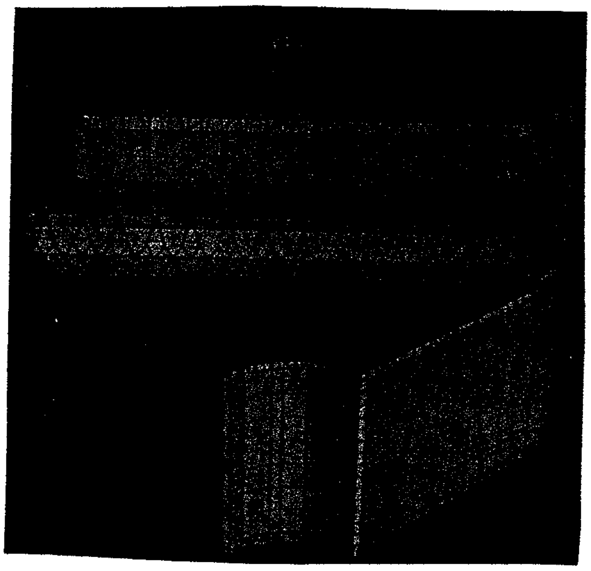
Table 1: Face octrees generation. Case study



a)



b)



c)

Figure 9: Case study: Complex solid.

6 Conclusion and Work in Progress

The GDM technique for extracting solid closed models from data generated by sensing and scanning technology was introduced in [18]. It has been used as the foundation for our face octree construction method. Given a voxel-based model, the algorithm presented generates a face octree with a high degree of compactation while preserving the degree of approximation of the initial data. We demonstrated the use of this algorithm in several practical exemples.

Even with the most efficient algorithms, extracting surfaces from large sets of volume data is always a time-consuming problem. In order to reduce the computation time, a parallel version of the presentred algorithm is now being programmed on a CM-2 SIMD machine. This is part of a larger project comprising parallelization of boolean operations between solids and surfaces both represented as face octrees.

7 Acknowledgements

We would like to sincerely thank Pere Brunet for his unvaluable help and for the fruitful ideas he suggested during the development of this work.

References

[1] E. Artzy, G. Frieder, and G. T. Herman. The theory, design, implementation and evaluation of a three-dimensional surface detection algorithm. *Computer Graphics and Image Processing*, 15:1–24, 1981.

[2] D. Ayala, P. Brunet, R. Juan, and I. Navazo. Object representation by means of nonminimal division of quadtrees and octrees. *ACM Transactions on Computer Graphics*, 4:41–59, 1985.

[3] P. Brunet. Face octrees: Involved algorithms and applications. Technical Report LSI–90–14, Universitat Politècnica de Catalunya, Department of LiSI, 1990.

[4] P. Brunet, I. Navazo, and A. Vinacua. A modelling scheme for the approximate representation of closed surfaces. *Computing*, 8:75–90, 1993.

[5] I. Carlbom, I. Chakravarty, and D. Vanderschel. A hierarchical data structure for representing the spatial decomposition of 3-D objects. *IEEE Computer Graphics and Applications*, pages 24–31, April 1985.

[6] H.E. Cline, W.E. Lorensen, S. Ludke, C.R. Crawford, and B.C. Teeter. Two algorithms for the three-dimensional reconstruction of tomograms. *Medical Physics*, 15(3):320–327, 1988.

[7] R.A. Drebin, L. Carpenter, and P. Hanrahan. Volume rendering. *Computer Graphics*, 2(4):65–74, 1988.

[8] M.J. Durst and T.L. Kunii. Integrated polytrees: A generalized model for integarting spatial decomposition and boundary representation. In *Theory and Practice of Solid Modeling*, Springer Verlag, 1989.

[9] H. Fuchs, Z.M. Kedem, and S.P. Uselton. Optimal surface reconstruction from planar contours. *Communications of the ACM*, 20(10):693–702, 1977.

[10] P.E. Gill, W. Murray, and M.H. Wright. *Practical Optimization.* Academic Press Inc., London, 1981.

[11] G.T. Hermann and H.K. Liu. Three-dimensional display of human organs from computer tomograms. *Computer Graphics and Image Processing*, 9, (1):1–21, 1979.

[12] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active contour models. *International Journal of Computer Vision*, pages 321–331, 1988.

[13] W.C. Lin, S.Y. Chen, and C.T. Chen. A new surface interpolation technique for reconstructing 3D objects from serial cross-sections. *Computer Vision, Graphics, and Image processing*, 48:124–143, 1989.

[14] W.E. Lorensen and H.E. Cline. Marching cubes: A high resolution 3D surface construction algorithm. *Computer Graphics*, 21(4):163–169, 1987.

[15] D.G. Luenberger. *Introduction to Linear and Nonlinear Programming.* Addison-Wesley Pu. Co., Reading, MA, 1973.

[16] B.H. McCormick, T.A. De Fanti, and M.D. Brown (Eds). Visualization in scientific computing. *Computer Graphics*, 21:1–21, 1987.

[17] D. Meagher. Efficient synthetic image generation of arbitrary 3D objects. In *IEEE Computer Society Conf. on Pattern Rec. and Image processing*, pages 473–478, 1982.

[18] J.V. Miller, D.E. Breen, W.E. Lorensen, R.M. O'Bara, and M.J. Wozny. Geometrically deformed models: A method for extracting closed geometric models from volume data. *Computer Graphics*, 25, 4:217–226, 1991.

[19] J.V. Miller, D.E. Breen, and M.J. Wozny. Extracting geometric models through constraint minimization. In *Visualization'90*, pages 74–82, 1990.

[20] M.A. Oliver and N.E. Wiseman. Operations on quadtree encoded images. *Computer Journal*, 26, 1:83–91, 1983.

[21] N. Plà-Garcia. Recovering a smooth boundary representation from an edge quadtree. In *Eurographics Workshop on Computer Graphics and Mathematics*, Genova, Italy, 1991.

[22] N. Plà-Garcia. Boolean operations and spatial complexity of face octrees. In R. Hubbold and R. Juan, editors, *Eurographics'93*, Barcelona, 1993.

[23] N. Plà-Garcia. *Techniques for modeling solids with smooth boundary.* PhD thesis, Universitat Politècnica de Catalunya, Computer Science, 1993. (In preparation).

[24] A. Requicha. Representations for rigid solids: Theory, methods, and systems. *Computing Surveys of the ACM*, 12:437–464, 1980.

[25] ———. NAG Fortran Library Manual. Mark 15. Technical report, The Numerical Algorithms Group Limited, 1991.

[26] H. Samet. *Applications of Spatial Data Structures.* Addison Wesley Publ., Reading, MA, 1989.

[27] H. Samet. *The Design and Analysis of Spatial Data Structures.* Addison Wesley Publ., Reading, MA, 1989.

[28] J. Serra. *Image Analysis and Mathematical Morphology*, volume 1. Academic Press., London, 1982.

[29] S. R. Sternberg. Grayscale morphology. *Computer Vision, Graphics, and Image Pprocessing*, 35:333–355, 1986.

[30] D. Terzopoulos, J. Platt, A. Barr, and K. Fleischer. Elastically deformable models. *Computer Graphics*, 21, 4:205–214, 1987.

[31] A. Witkin, K. Fleischer, and A. Barr. Energy constraints on parameterized models. *Computer Graphics*, 21, 4:225–232, 1987.

Departament de Llengatges i Sistemes Informàtics
Universitat Politècnica de Catalunya

List of research reports (1993).

LSI-93-1-R "A methodology for semantically enriching interoperable databases", Malú Castellanos.

LSI-93-2-R "Extraction of data dependencies", Malú Castellanos and Fèlix Saltor.

LSI-93-3-R "The use of visibility coherence for radiosity computation", X. Pueyo.

LSI-93-4-R "An integral geometry based method for fast form-factor computation", Mateu Sbert.

LSI-93-5-R "Temporal coherence in progressive radiosity", D. Tost and X. Pueyo.

LSI-93-6-R "Multilevel use of coherence for complex radiosity environments", Josep Vilaplana and Xavier Pueyo.

LSI-93-7-R "A characterization of $PF^{NP\|} = PF^{NP[\log]}$", Antoni Lozano.

LSI-93-8-R "Computing functions with parallel queries to NP", Birgit Jenner and Jacobo Torán.

LSI-93-9-R "Simple LPO-constraint solving methods", Robert Nieuwenhuis.

LSI-93-10-R "Parallel approximation schemes for problems on planar graphs", Josep Díaz, María J. Serna, and Jacobo Torán.

LSI-93-11-R "Parallel update and search in skip lists", Joaquim Gabarró, Conrado Martínez, and Xavier Messeguer.

LSI-93-12-R "On the power of equivalence queries", Ricard Gavaldà.

LSI-93-13-R "On the learnability of output-DFA: a proof and an implementation", Carlos Domingo and David Guijarro.

LSI-93-14-R "A heuristic search approach to reduction of connections for multiple-bus organization", Patricia Ávila.

LSI-93-15-R "Toward a distributed network of intelligent substation alarm processors", Patricia Ávila.

LSI-93-16-R "The Odissea approach to the design of information systems from deductive conceptual models", Maria Ribera Sancho and Antoni Olivé.

LSI-93-17-R "Constructing face octrees from voxel-based volume representations", Robert Juan i Ariño and Jaume Solé i Bosquet.