Meshes

Finite Element Meshes by Means of Voxels

Pascal J. FREY* and Houman BOROUCHAKI

INRIA, Gamma project,
Domaine de Voluceau, Rocquencourt,
BP 105, 78153 Le Chesnay Cedex, France.

Abstract. A number of techniques have been developed for the polygonization of implicit surfaces or isosurfaces. Two approaches for generating polyhedral approximations of implicit models are presented. The first method produces finite element meshes. The second algorithm is based on a $P1$ (linear) surface approximation from volumetric datasets. Applications examples of discrete constructive solid geometry objects are given.

Key-words. Implicit models, Surface reconstruction, Polygonization, Finite element mesh, Voxel, Texel.

1 Introduction

This paper is concerned with the polyhedrization of implicit objects. Implicit surfaces and volumes arise frequently in CAD applications. Typically, they are the result of Constructive Solid Geometry (CSG) operations, and complex geometries can be defined in terms of boolean operations. On the other hand, the availability of fast, accurate scanning devices provide sampled volumetric data for applications such as medicine and engineering. In this context, the problem of isosurface extraction is closely related to the general implicit surface reconstruction problem.

Problem statement

Recent years have witnessed an increasing interest in the use of implicit functions for defining geometric objects in the field of Computer-aided geometric design (CAGD). They are called "implicit" because they represents subsets of E^3 that are not specified explicitely by their boundaries or parametrizations. An implicit algebraic surface is given by [9]

$$f(x,y,z) = \sum_{i,j,k} a_{ijk} x^i y^j z^k = 0 \tag{1}$$

where f is a polynomial in x, y and z and a_{ijk} are numbers. The surface consists of all real or complex points (x,y,z) that satisfy the equation. A geometric object is considered as a closed subset of E^3 with the definition $f(x,y,z) \geq 0$ and called a *solid*. The boundary of such an object is a so-called implicit surface. The *defining function* f

* E-mail: Pascal.Frey@inria.fr

may be defined procedurally, *i.e.* encoded by an algorithm that returns some (usually heuristically obtained) value or approximated by data sets. There is a classification of points in E^3 with respect to the solid, if $P = (x, y, z)$ is a point of E^3, then

$$\begin{array}{ll} f(P) > 0 & \text{if } P \text{ is inside the object;} \\ f(P) = 0 & \text{if } P \text{ is on the boundary;} \\ f(P) < 0 & \text{if } P \text{ is outside.} \end{array} \tag{2}$$

Several algorithms of polygonization of implicitly defined surfaces or isosurfaces of trivariate functions have been proposed. Similar algorithms were introduced for extracting polygonal surfaces from volumetric data sets, which, in fact, use tabulated functions of three variables. No preliminary information about the topology of the object is required and only cartesian coordinates of points are used in the reconstruction. The topology of an implicit surface refers to the number of disjoint components together with the *genus* (number of holes) of each component. Thus, a topologically correct polygonization must share the same number of components, each component having the same genus, as its corresponding component in the original surface [26].

We were led to consider the general surface reconstruction problem stated above by a number of application areas in science and engineering, including finite element modeling, that require interpolation or smoothing of large arrays of scattered points of a surface. The main sources of such data are physical measurements taken by non-invasive sensing and scanning devices. The generation of a finite element mesh is the main bottleneck in the finite element modeling process. Conventional approaches require a continuous analytic solid model, rather than a discrete solid model. A leading technique for constructing isosurfaces from sampled data is to consider cells with sample points as corner and to approximate the surface location between data points by polygons. However, this technique leads to an excessively large number of polygons to represent an isosurface.

This paper presents a general-purpose volumetric method for generating tetrahedral meshes from scattered data, which combines a divide-and-conquer isosurface tracking and tetrahedral element generation. The proposed algorithm uses adaptive downsampling (*octree*) as a way to reduce the number of mesh elements. After describing this method, we introduce a new technique for generating triangular-quadrilateral isosurface meshes. Results from different modeling applications illustrate the strength of the algorithms. As the title of the paper suggests, we are primarily concerned with creating geometrically accurate meshes to be used in finite element simulations. Therefore, we review briefly the different classes of methods capable of tessellating sampled data.

Related work

In comparison with the literature on parametric surfaces, there are relatively few works on implicitely defined surfaces. Half spaces defined by algebraic inequalities are commonly used as primitives in CSG [19]. Ricci [20] presented a general approach to the definition of complex objects from simpler ones using constructive geometry. Wyvill et al. [29] described an approach for modeling, visualizing and animating so-called *soft* objects, defined as a level surface of a set of given field functions. Bloomenthal [2] discussed a numerical technique for constructing an adaptive polygonization of an implicit

function using octree-based space partitioning. Lorensen and Cline [11] described an algorithm (*Marching Cubes*, MC) for constructing a polygonal representation of a constant density surface using voxels and numerous algorithms for guaranteeing topological correctness of the polygonization of isosurface have been proposed (Matveyev [12], Wilhelms and Van Gelder [28], Stander and Hart [26]). Models deformations and displacement maps (Sclaroff and Pentland [23]), as well as polynomial functions (Bajaj et al [1]) have also been applied for solving the surface reconstruction problem. Thus, function representation is widely used in geometric modeling and computer graphics in several forms. However, most methods expect the surface to be regular, because they rely on the partioning algorithm that can always be made small enough to yield to an unambiguous intersection with the surface. Frey et al. [6] introduced a finite element mesh generation algorithm suitable for objects based on sampled data. This technique extracts a closed, consistent and continuous polyhedral isosurface from the sampled data by processing the boundary voxels only.

The motivation of our work is to develop an automatic tetrahedral mesh generation algorithm dealing with discrete objects. The main objective is to control the accuracy of the polyhedral approximation of the geometric model, in particular, the surface mesh adequacy to finite elements requirements [7].

Outline

This paper is organized as follows. In the second section a short review of the background material for implicit objects definition and voxels representation is given along with numerical algorithms. The third section provides the details of the voxel-based mesh generation technique. The fourth section introduces the Texel algorithm for accurate isosurface approximation and provides information about the possible extension to tetrahedral mesh generation. Future works are mentioned in the last section.

2 A discrete constructive solid geometry

This section provides a brief summary of geometric concepts of an implicit function-based modeling environment. In principle, an implicit surface is the zero-set of a valued function, which can be either a polynomial, a transcendental or a procedurally defined function. The only requirement for this function to obtain continuous surfaces is to be continuous. A polygonal approximation can be derived from the implicit surface which can be computed effciently using adaptive subdivision techniques. Thus, complicated surfaces can be studied without the need to solve the implicit function. The transformations of the function representation associated with operations on an object are reviewed in this section.

2.1 Set-theoretic operations

Many interesting properties of R-functions have been studied by Rvachev for solving problems of mathematical physics in areas of complex shapes [21]. He defined how to transform operations on areas described by equation (2) to operations on R-functions.

Thus, one of the possible analytical descriptions of the R-functions *union* and *intersection* is

$$f_1 \wedge f_2 = \frac{1}{1+\alpha}\left(f_1 + f_2 - \sqrt{(f_1^2 + f_2^2 - 2\alpha f_1 f_2)}\right)$$
$$f_1 \vee f_2 = \frac{1}{1+\alpha}\left(f_1 + f_2 + \sqrt{(f_1^2 + f_2^2 - 2\alpha f_1 f_2)}\right) \tag{3}$$

where f_1 and f_2 are defining functions of initial objects and $\alpha = \alpha(f_1, f_2)$ is an arbitrary continuous function satisfying the following conditions

$$-1 < \alpha(f_1, f_2) \leq 1\,,$$
$$\alpha(f_1, f_2) = \alpha(f_2, f_1) = \alpha(-f_1, f_2) = \alpha(f_1, -f_2)$$

The expression for the subtraction operation is $f_1 \backslash f_2 = f_1 \wedge (-f_2)$. Notice that with this definition, the resulting object includes its boundary. If $\alpha = 1$ equations (3) become (cf. Ricci [20])

$$f_1 \wedge f_2 = \min(f_1, f_2)$$
$$f_1 \vee f_2 = \max(f_1, f_2)\,. \tag{4}$$

A major drawback of this description is that the equations (3) have C^1 discontinuity when $f_1 = f_2$. A most useful form providing C^m continuity uses the following set of R-functions:

$$f_1 \wedge f_2 = (f_1 + f_2 - \sqrt{f_1^2 + f_2^2})(f_1^2 + f_2^2)^{\frac{m}{2}}$$
$$f_1 \vee f_2 = (f_1 + f_2 + \sqrt{f_1^2 + f_2^2})(f_1^2 + f_2^2)^{\frac{m}{2}} \tag{5}$$

Note that R-functions can be used to describe geometric primitives. Pasko et al. [17] described transformations of a defining real function for set-theoretic operations, blending, offsetting, bijective mapping, projection, cartesian products and metamorphosis, along with inclusion, point membership and intersection relations.

2.2 d_{CSG}-objects

As explained in the previous section, geometric objects can be given by their defining functions. Pasko et al. [17] described a high-level geometric language that can extend the interactive modeling system based on function representation notion. Here, we give an example of an application of geometric modeling to construct so-called d_{CSG}-objects. Figure 1 shows the polygonization of a discrete sphere truncated by a unit cube, sampled using a $32 \times 32 \times 32$ grid. Figure 3 illustrates the combination of d_{CSG}-primitives (cylinders, spheres) to form a more complex d_{CSG}-object. A simple isosurface approximation is shown on figure 4; the sampling used a $20 \times 20 \times 20$ grid.

In this section, we have explained how geometric models can be defined with defining functions. In the next section, we will describe two mesh generation methods based on implicit function representation.

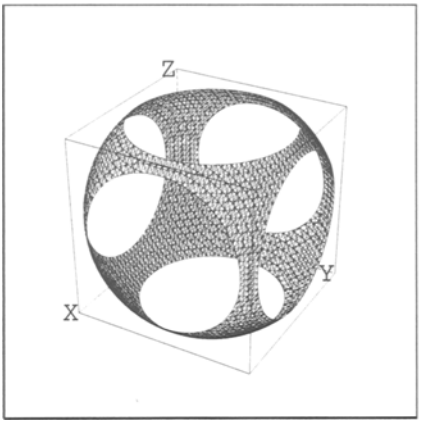

Fig. 1. *truncated d_{CSG}-sphere*

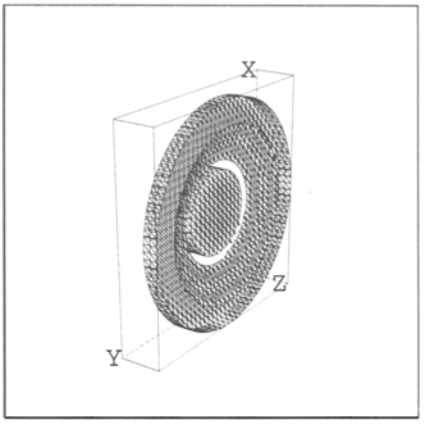

Fig. 2. *d_{CSG}-object*

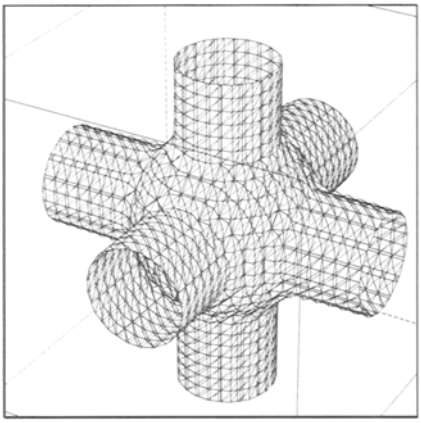

Fig. 3. *d_{CSG}-model: union of cylinders and sphere*

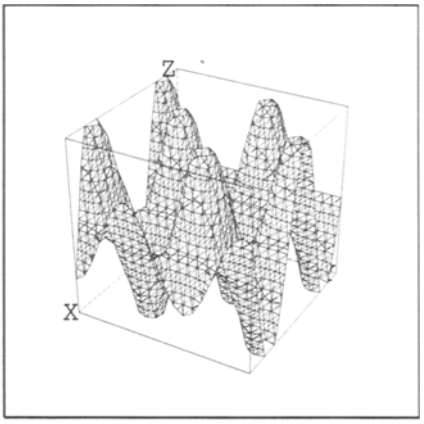

Fig. 4. *implicit surface* $f(x, y, z) = z - 3\sin(2x)\cos(2y)$

3 Voxel-based tetrahedralization

The basic idea of all methods for creating a polyhedral approximation of an object is an appropriate subdivision of the relevant space. Polygonization algorithms typically query the implicit surface through spatial sampling. The space is subdivided into small cells, each of which yields a piece of the desired approximation [22]. The combination of these cells provides the approximation of the whole object. In the case of voxel-based objects, the basic cell is a hexahedron. The decision of whether a voxel is intersected by the surface can be established by looking at the function values $f(P)$ at the eight vertices of the hexahedron. The surface intersects the voxel if not all signs of these values are equal. The coordinates of the points of voxel edges intersecting the isosurface are computed using a linear interpolation.

3.1 Voxel Mesh Generation

The voxel-based mesh algorithm ($\mathcal{VMA}$) uses an iterative procedure to create tetrahedral elements in each voxel of the d_{CSG}-solid representation. A two-steps approach is required to mesh the model. First, the voxels are analyzed and intersection points are computed. Then, tetrahedral elements are created. During the subdivision process, three types of voxels can be found:

- those disjunct from the d_{CSG}-solid;
- those intersecting the boundary of the d_{CSG}-solid ;
- those containing a piece of the d_{CSG}-solid (eight voxel vertices satisfying the defining function).

These voxels are classified as $\mathcal{O}$ (outside), $\mathcal{B}$ (boundary), or $\mathcal{I}$ (interior), respectively. From the above definition, it appears that $2^8 = 256$ configurations can be encountered during the subdivision procedure. However, in virtue of topological invariability through symmetry and rotation, there are only a few possible cases of intersection of the solid and a voxel [6]. This observation leads to implement a minimal set of patterns providing the tetrahedralization of a given voxel. Numerically, the voxel identification can be based on vertex numbering using a binary coding [11].

Although the principle stated before appears to be suitable for finite element mesh generation, in particular the mesh conformity is ensured, it can lead to erroneous topology and possible *holes* in the domain as a result of the incorrect voxel truncature. A simple solution to the *bubble problem* has been proposed by Frey et al. [6]. The voxel approach is obviously well-suited for meshing the space surrounding a d_{CSG}-object. $\mathcal{VMA}$ algorithm can also extract a closed surface from the volumetric data by looking at the $\mathcal{B}$ voxels. The resulting triangular mesh is a regular surface of class C^2, with no holes.

Uniform partitioning, such as $\mathcal{VMA}$, has the advantage of being easy to implement. Two fundamental drawbacks, however, are that an excessively large number of tetrahedra is produced and no mesh gradation can be achieved. This problem is compounded by the algorithm itself, since each mesh face lies within a single voxel. In order to make the process of meshing a d_{CSG}-model more efficient, adaptive downsampling and space partitioning (based on an octree) are used as a way to reduce the number of mesh elements.

3.2 Adaptive partitioning

The octree-based mesh generation method involves three steps. The first step is *surface tracking*, which is the application of the $\mathcal{VMA}$ to the dataset. The result of this step is stored in an octree data structure. The depth of the octree is determined from the largest dimension l_{max} of the dataset as $log(l_{max})$. During this process, all voxels marked $\mathcal{I}$ and $\mathcal{B}$ are identified and stored as leaf nodes. At any level of the octree, the algorithm attempts to replace eight child octants with a single parent octant, level by level from bottom to top. The second step consists in *patching* the octree. Patching strategy is achieved by traversing the octree in a breadth-first manner and forces octants that share one edge to have no more than one level difference. The one-level difference rule is commonly used in octree-based meshing procedures to control mesh gradations and element aspect ratios. The third step of the procedure is the creation of the tetrahedral elements that make up the mesh.

The octree is a convenient technique for adaptively storing information about the d_{CSG}-model. As outlined by Bloomenthal [2], the partitioning space provides a framework for connecting surface points into a polyhedral representation. Moreover, the application of octree technique to unstructured mesh generation has nearly a decade history [24].

3.3 Application examples

In this section, an example of a polygonized object using the above methods is explored, and several characteristics of the methods are illustrated. These methods ae indifferent to the topology of the d_{CSG}-object. Given an implicit surface that is continuous and closed, Euler's formula can be used to compute the genus of the surface. Figure 5 shows the mesh of a d_{CSG}-object obtained using $\mathcal{VMA}$. Figure 6 shows the polyhedral representation of the same model using the octree-based method. This mesh consists of 32228 tetrahedral elements and has been generated in less than 2 seconds of CPU time (HP 735/99Mhz).

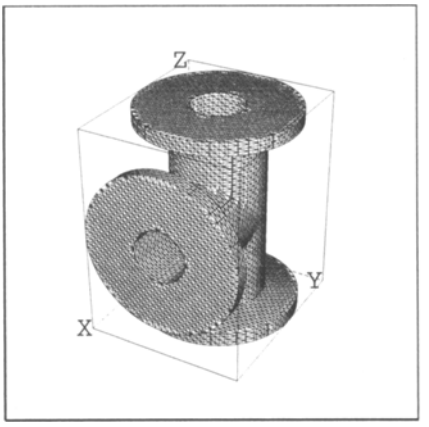

Fig. 5. *Voxel-based mesh*

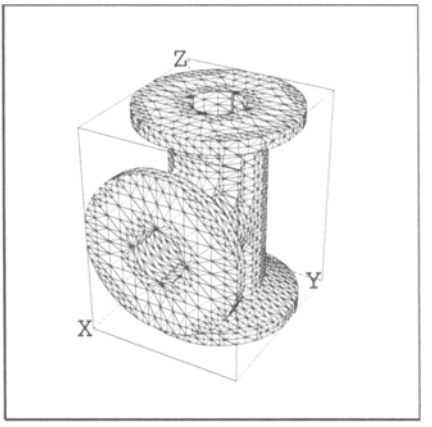

Fig. 6. *Octree-based mesh*

The examples presented so far were related to d_{CSG}-objects. The authors tried the algorithm on biomedical surfaces defined from from computer tomography imaging (CTI) and magnetic resonance imaging (MRI) data sets, leading to meshes with more than one million elements.

In numerical simulations and geometric modeling the goal is to find an optimal piecewise planar approximation of the original surface so that the maximal distance between the original and the approximating surface does not exceed a given tolerance. In the context of the finite element methods, the quality of the geometric approximation is of significant importance because of its effect on the accuracy of the numerical solutions and the convergence of the computation scheme. Therefore, we propose a new algorithm for isosurface approximation, which ensures a $P1$ approximation of the given surface.

4 Texel algorithm

In this section, we introduce a new method for isosurface approximation, resulting in mixed (triangular-quadrilateral) meshes. In order to obtain a pure triangular mesh, we will also propose a solution to split a quadrilateral mesh face into two triangles, with respect to geometric criteria [7]. The main purpose of this approach is to overcome the problem of ambiguity over the faces arising in MC-like algorithms and to offer a more accurate surface approximation ($P1$-type, linear) with improved efficiency (in terms of CPU time). The method present some similarities with the surface mapping function described by Payne and Toga [18].

The method consists in considering a tetrahedral voxel partition which will be used for linear interpolation. Similar to MC approach, the voxels are analyzed and a tracking algorithm determines the isosurface intersection with each tetrahedral cell. This results in only two types of surface patterns, triangles and quadrilaterals. Mesh conformity is automatically ensured and the problem of ambiguity is not encountered due to the tetrahedral partitioning space. By nature, this method provides $P1$ interpolation for isosurface approximation which has proven to be more reliable than the conventional MC procedure [8].
Figure 7 shows an isosurface approximation with the texel algorithm. The analytic surface is defined by

$$f(x, y, z) = z - x^2 - xy\,.$$

The isosurface reconstruction produced 2570 vertices and 3761 in 0.89 second (including the IO on a HP 735/99). Figure 8 represents the subtraction of a cylinder from a sphere with respect to a $32 \times 32 \times 32$ grid. The isosurface polygonal representation, obtained in 3 seconds, contains 7532 vertices and 11624 faces (8660 triangles).

This algorithm is also well suited for tetrahedral mesh generation of implicitly defined objects. Indeed, each intersected tetrahedral cell is triangulated with respect to the isosurface intersection, in such a way as to ensure the mesh conformity with the neighbouring cells [3]. A significant drawback of this method is that, compared to MC-like algorithms, the resulting meshes have more vertices. Thus, the issue of the decimation of large meshes is of utmost importance in the context of numerical simulations.

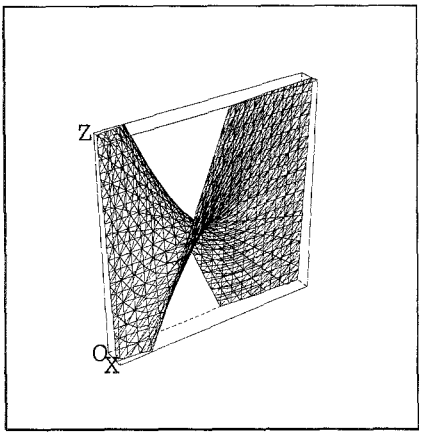

Fig. 7. *isosurface*

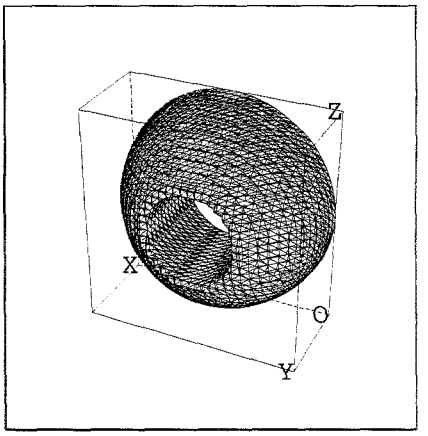

Fig. 8. *extrusion of a d_{CSG}-sphere*

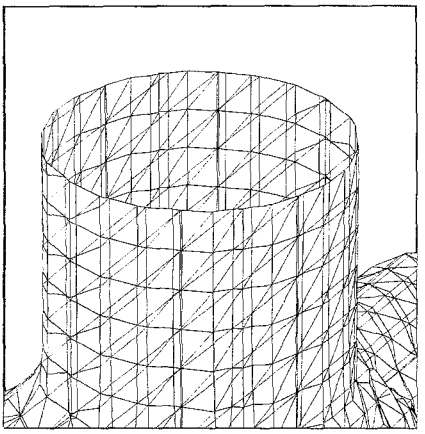

Fig. 9. *d_{CSG}-model: partial zoom*

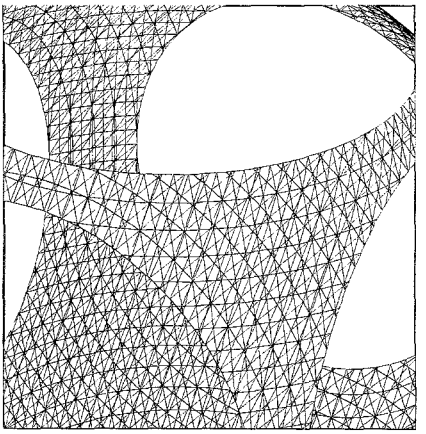

Fig. 10. *d_{CSG}-sphere: partial zoom*

5 Future work

We have presented two algorithms for generating a polyhedral mesh from a given implicitly defined model. The implicit definition of geometric objects offers numerous advantages, in particular an abstraction level higher than those of others representations. Moreover, objects defined in spaces of various dimensions are uniformly represented. The ability to derive a polyhedral representation from an implicitly defined model has numerous consequences for the modeling process. One main interest is that the method isolates the polyhedral representation from the complexities of the model. With this approach complex shapes can be easily defined.

The proposed algorithms may be supplemented or improved in many ways. A number of issues await further investigation. For instance, the octree has proven a convenient mechanism for adaptively storing information about the surface and its partitioning space is a clever framework for the connectivity of the polygonal approximation. Octree will probably lead to significant improvements in performance and efficiency for

tetrahedral reconstruction. In particular, the texel algorithm shall use an octree representation during the tetrahedralization step. This topic needs to be explored in order to produce meshes suitable for finite element methods.

With the current surface approximation algorithms, often thousands or millions of primitives are required to capture the details of complex models. Consequently, applications need algorithms that use local operations on geometry and topology to reduce the number of triangles in the meshes. These decimation techniques guarantee that a simplified mesh approximates the original mesh to within a prespecified tolerance. We are currently working on a decimation algorithm that uses geometric criteria to control the removal of mesh features.

References

1. C. Bajaj, I. Ihm and J. Warren, Higher-order interpolation and least-squares approximation using implicit algebraic surfaces, *ACM Trans. Graph.*, vol 12, pp. 327–347, 1993.
2. J. Bloomenthal, Polygonization of implicit surfaces, *CAGD*, vol 5, pp. 341–355, 1988.
3. H. Borouchaki, Triangulation sous contraintes en dimension quelconque, Rapport de Recherche INRIA, RR-2373, 1994.
4. B.K. Choi, H.Y. Shin and J.W. Lee, Triangulation of scattered data in 3D space, *CAD*, vol 20, no 5, pp. 239–248, 1988.
5. M.J. Dürst, Letters: additional reference to "Marching Cubes", ACM Comp. Graphics, vol 22, no 4, pp. 72–73, 1988.
6. P.J. Frey, B. Sarter and M. Gautherie, Fully automatic mesh generation for 3D domains based upon voxel sets, *Int. J. Numer. Meth. Engng.*, vol 37, pp. 2735–2753, 1994.
7. P.J. Frey and H. Borouchaki, Geometric evaluation of finite element surface meshes, submitted to *Comput. Aided Geom. Design*
8. P.J. Frey and H. Borouchaki, Texel meshing, to appear.
9. c. M. Hoffmann, Implicit curves and surfaces in CAGD, *IEEE Comp. Graphics Appl.*, vol 13, pp. 79–88, 1993.
10. H. Hoppe, Surface reconstruction from unorganized points, Phd thesis, Univ. of Washington, 1994.
11. W.E. Lorensen and H.E. Cline, Marching cubes: a high-resolution 3D surface construction algorithm, Siggraph'87 Conf. Proc., *Comp. Graphics*, vol 21, no 4, pp. 163–169, 1987.
12. S.V. Matveyev, Approximation of isosurface in the Marching Cube: ambiguity problem, Visualization'94, Conf. Proc., *IEEE Computer Society Press*, pp. 288–292, 1994.
13. C. Montani, R. Scateni and R. Scopigno, A modified look-up table for implicit disambiguation of Marching Cubes, *The Visual Computer*, vol 10, 1994.
14. H. Muller and M. Stark, Adaptive generation of surfaces in volume data, *The Visual Computer*, vol 9, no 4, pp. 182–199, 1993.
15. The asymptotic decider: resolving the ambiguity in Marching Cubes, Visualization'91, Conf. Proc., *IEEE Computer Society Press*, pp. 83–90, 1991.
16. P. Ning and J. Bloomenthal, An evaluation of implicit surface tilers, *IEEE Computer Graphics & Applications*, vol 13, no 6, pp. 33–41, 1993.

17. A. PASKI, V. ADZHIEV, A. SOURIN AND V. SAVCHENKO, Function representation in geometric modeling: concepts, implementation and applications, *The Visual Computer*, vol 11, pp. 429–446, 1995.
18. B. PAYNE AND A. TOGA, Surface mapping brain function on 3d models, *IEEE Computer Graphics & Applications*, 1990.
19. A.A.G. REQUICHA, Representations for rigid solids: theory, methods and systems, *Comput. Surveys*, vol 12, pp. 437–464, 1980.
20. A. RICCI, A constructive geometry for computer graphics, *The Computer Journal*, vol 16, pp. 157–160, 1972.
21. V.L. RVACHEV, On the analytical description of some geometric objects, Report of the Ukrainian Academy of Sciences, vol 153, pp. 765–767, 1963.
22. M.F.W. SCHMIDT, Cutting cubes - visualizing implicit surfaces by adaptive polygonization, *The Visual Computer*, vol 10, pp. 101–115, 1993.
23. S. SCLAROFF AND A. PENTLAND, generalized implicit functions for computer graphics, *Comput. Graph.*, vol 25, pp. 247–250, 1991.
24. M.S. SHEPHARD AND M.K. GEORGES, Automatic three-dimensional mesh generation by the finite octree technique., *Int. J. Numer. Methods Eng.*, vol 32, pp. 709–749, 1991.
25. W.J. SCHROEDER, J.A. ZARGE AND W. LORENSEN, Decimation of triangle mesh, *ACM Comput. Graphics*, vol 26, no 2, pp. 65–70, 1992.
26. B.T. STANDER AND J.C. HART, Guaranteeing the topology of an implicit surface polygonization, to appear in Siggraph'96 Conf. Proc., *Comput. Graphics*, 1996.
27. G. TURK, Re-tiling polygonal surfaces,*Comput. Graphics*, vol 26, no 2,pp. 55–64, 1992.
28. J. WILHELMS AND A. VAN GELDER, Topological considerations in isosurface generation extended abstract, *Comput. Graphics*, vol 24, no 5, pp. 79-86, 1990.
29. G. WYVILL, C. MCPHEETERS AND B. WYVILL, Data structure for soft objects, *The Visual Computer*, vol 2, pp. 227–234, 1986.
30. R. YAGEL, D. COHEN AND A. KAUFMAN, Normal estimation in 3D discrete space, *The Visual Computer*, vol 8, pp. 278–291, 1992.