Shape reconstruction from planar cross-sections

Jean-Daniel Boissonnat

CENTRE DE ROCQUENCOURT

Institut National
de Recherche
en Informatique
et en Automatique

Domaine de Voluceau
Rocquencourt
B.P.105
78153 Le Chesnay Cedex
France
Tél.:(1) 39 63 55 11

Rapports de Recherche

N° 546

SHAPE RECONSTRUCTION FROM PLANAR CROSS-SECTIONS

Jean-Daniel BOISSONNAT

Juillet 1986

RESUME :

On propose une solution originale au problème de la reconstruction de formes tridimensionnelles à partir de sections planes. Le principe consiste à calculer un volume dont le bord est un polyèdre à facettes triangulaires s'appuyant sur les contours de départ. Ce volume est obtenu, tranche par tranche, à partir de la triangulation de Delaunay associé à deux sections adjacentes. On montre que la triangulation de Delaunay peut être calculée de façon efficace dans cette situation particulière : on fournit un algorithme optimal en fonction de l'entrée et de la sortie, ne faisant intervenir que des opérations 2d. La méthode de reconstruction permet d'extraire de cette triangulation un solide maximal ne comportant pas de singularité, et résoud le cas général où le nombre de contours varie d'une coupe à l'autre (branchements, trous, ...) . La méthode a été implémentée et testée sur de nombreux exemples ; des résultats sur des images médicales sont présentés.

Mots clés : reconstruction de formes 3d, tomographie, géométrie algorithmique, triangulation de Delaunay, représentation de formes 3d, appariements de formes.

June 1986

SHAPE RECONSTRUCTION FROM PLANAR CROSS-SECTIONS

Jean-Daniel Boissonnat

INRIA

Domaine de Voluceau

Rocquencourt BP 105

78153 LE CHESNAY FRANCE

ABSTRACT

We propose a new solution to the problem of constructing a shape over a set of L cross-sectional contours. We construct a volume whose boundary is a polyhedron with triangular faces intersecting the cutting planes along the given contours. This volume is obtained by pruning L-1 Delaunay triangulations formed between adjacent cross-sections. The Delaunay triangulation can be computed efficiently in this specific situation and the method can handle the case where there are several connected components in each cross section. This algorithm has been fully implemented and tested on various examples; results on medical images are presented.

Key-words: 3-d shape reconstruction, tomography, computational geometry, Delaunay triangulation, shape representation, shape matching.

1- INTRODUCTION

In many applications, an object or a set of objects are known by a sequence of cross-sections. For instance, in medical diagnosis and therapy, cross-sectional images are obtained by moving an echographic or an X-ray apparatus. The reconstructed structure ther appears similar to a set of wire-frame contours over which the surface "skin" is thought to lie (see Fig. 20). Since the display of the structure is most often in the form of these wire-frame contours, the task of creating the surface or of creating another cross-section is left to the viewer's imagination. This is a non-trivial mental exercise when dealing with any but the simplest structures.

Several solutions have been proposed in the literature. We can distinguish two approaches. In the first one, the data consist in a 3-d image made of voxels. The problem is then to extract and follow the faces of the voxels which are on the boundary of the objects. This is a typical situation when dealing with CT images which has been extensively studied by Herman and his coworkers (see for example (AFH)). However, in many other situations, we do not have any 3-d structure (such as voxels): the information about the 3-d objects consist only of planar contours. The problem is not a detection problem but an interpolation problem. Keppel (K) and Fuchs et al. (FKU) reduce this problem to constructing a sequence of surfaces, one between each pair of adjacent contours. These surfaces are constructed solely from elementary triangular tiles, each defined between two consecutive points on the same contour and a single point on an adjacent contour. Constructing such a surface is shown to be equivalent to finding a path in a directed graph. Then if we associate a weight to the edges of that graph, a classical shortest path algorithm will produce an optimal reconstruction. Several possible weights have been proposed, based on the volume enclosed by the triangular stripe (K), its area (FKU) or the length of its edges linking the two contours (P). Another weight has been proposed by Ganapathy and Dennehyn, namely the length of the contour segment included in the tile, divided by the perimeter of the contour on which that tile lies (GD). The existence of many possible criteria made this method useful for a broad range of applications, though, as Sloan and Hrechanyk pointed out (SH), the tiling algorithms are not appropriate when there are major differences in shape or position between successive contours. An extreme case will be that of a convex contour in one plane and a of spiral-like contour in the other: the result will be a surface crossing itself. The main drawback, however, of this elegant method is that it cannot handle the important case of objects with multiple contours in a single plane. The reconstruction, for instance, of a standing human figure from contours on horizontal planes would involve perhaps three contours (two arms and the torso) at chest levels, merging into a single contour at the shoulder level. The only attempt in that direction seems to have been done by Shantz (Sh) but it is very heuristic and fail in many cases, as declared by the author himself.

This paper presents an entirely new approach (based on the ideas of the preliminary paper (B1)). Our method can handle the cases where the object contains multiple contours and where the number of contours varies from one cross-section to the other. Unlike the method of Fuchs et al., our method does not construct the surface of the object, but its volume. At the end of the procedure, it will be easy, of course, to produce the surface of the object by looking at the boundary of the obtained volume. This method uses the Delaunay triangulation and can be considered as a specific application of the general ideas of (B2). In (B2), the input data consisted of a given set of points on the boundary of an object but no a priori structure between the points was assumed to exist. Here, we take advantage of the additional knowledge we have; the neighboring relationships between the points belonging to the same cross-section is explicitly used, making possible to handle the case of multiple objects and the case of objects with holes.

The problem can be formulated in the following way. One or several unknown 3-dimensional objects are intersected by a finite number of specified, but not necessarily parallel, planes. The only information about the objects consists of the intersections of their surfaces with the planes. Each of these intersections can be represented by a finite number of oriented closed curves, separating the inside of the objects from the outside. These curves are approximated by closed polygons, also called contours in the sequel. We will note $\mathbb{C}$ the set of the contours and $\mathbb{M}$ the set of the vertices of $\mathbb{C}$. The sequence of contours is used to construct a polyhedral approximation to the original object. As in (FKU), the problem is reduced to constructing a sequence of polyhedral slices whose boundaries are made of triangular faces, one between each pair of adjacent contours.

The paper is organized as follows: in Section 2, we recall the definition and some general properties of the Delaunay triangulation, and derive new results in the case of interest here where the points lie on two planes. In particular, an algorithm computing the Delaunay triangulation is presented which is optimal with respect to the input and the output. Section 3 shows how to achieve a correct reconstruction of the original object. We conclude in Section 4 with some additional remarks and experimental results.

2- THE DELAUNAY TRIANGULATION

2-1 Definition and general results

Let us recall the definition of the Delaunay triangulation and some general results. For a d-dimensional Euclidean space $\mathbb{E}$ and a set $\mathbb{M}$ of N points $m_1 \ldots m_N$, the associated *Voronoi diagram* is a sequence $(V(m_1) \ldots V(m_N))$ of convex polyhedra covering $\mathbb{E}$ where $V(m_i)$ consists of all the points of $\mathbb{E}$ that have m_i as a nearest point in the set $\mathbb{M}$. Thus

$$V(m_i) = \{ P \in \mathbb{E}: \forall\ j,\ 1 \leq j \leq N,\ \delta(P,M_i) \leq \delta(P,M_j) \}$$

where δ denotes the Euclidean distance.

The geometrical dual of the Voronoi diagram, obtained by linking the points m_i whose Voronoi polyhedra are adjacent across a common face, is called the *Delaunay triangulation* of $\mathbb{M}$. While the dual graph is usually considered as an abstract graph expressing the topological relationships of the Voronoi polyhedra, we are here concerned to take the joins to be straight line segments, and to use them as a framework for a simplicial subdivision of space. Fig.1 shows an example of a Voronoi diagram and its dual in a simple 2-d case. It can be shown (R) that two elements are either disjoint or have exactly one k-simplex[1] (k<d) in common. Moreover, the union of the elements of the Delaunay triangulation is equal to the interior of the convex hull of $\mathbb{M}$. Due to the definition, when no set of d+2 points are cospherical, the elements of the Delaunay triangulation are d-simplices, and the circumspheres (the *Delaunay spheres*) do not contain any point of $\mathbb{M}$ in their interior. This property characterizes the Delaunay triangulation: if a sphere circumscribing d+1 points of $\mathbb{M}$ does not contain any other point of $\mathbb{M}$ in its interior, this sphere is a Delaunay sphere and the corresponding d+1 points belong to a simplex of the Delaunay triangulation. In case of

---

1: a k-simplex is a convex polytope with k+1 affinely independent vertices; a 2-simplex is a triangle, a 3-simplex is a tetrahedron.

more than d+2 cospherical points, the elements can be decomposed into several simplices, so that, in every case, the Delaunay triangulation is composed of simplices. It can be shown (B2) that the Delaunay triangulation is a d-connected graph on M embedded in $\mathbb{R}^3$, which defines symmetrical and isotropic neighborhood relationships between the points. It is to be noticed that this is not necessarily the case of the k-nearest neighbors or of other geometric structures included in the Delaunay triangulation such as the minimum spanning tree, the Gabriel graph or the neighborhood graph. Moreover a number of polyhedra are contained in the Delaunay triangulation, which is of particular interest for our purpose.

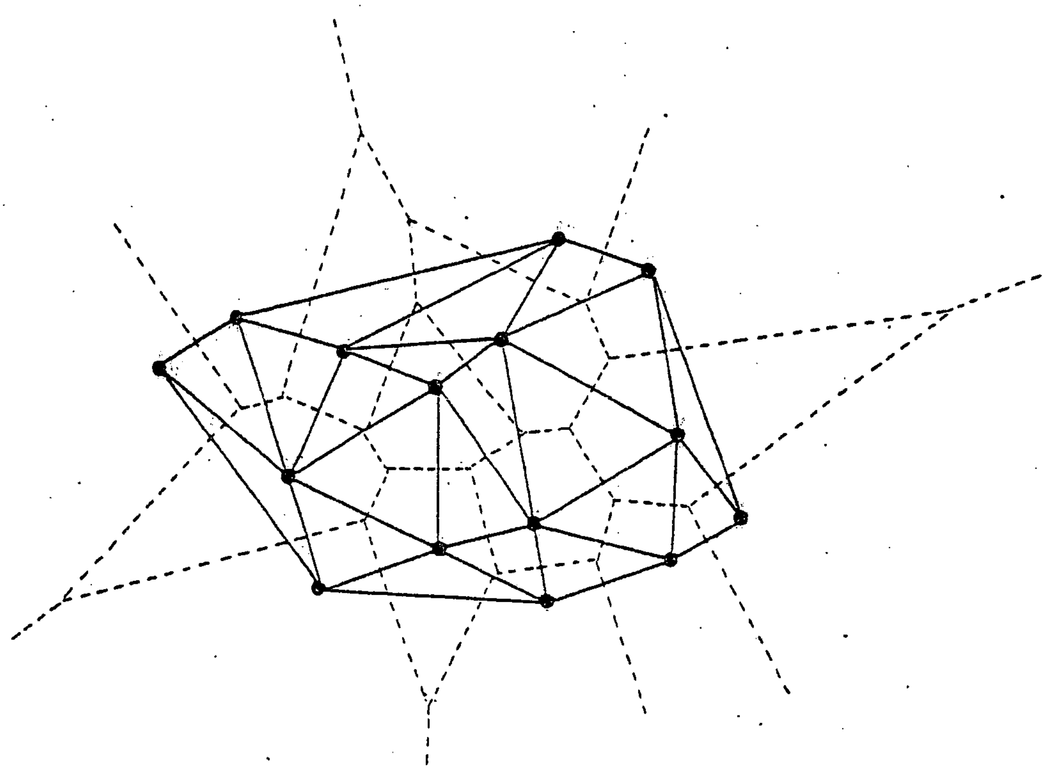


Figure 1

● Points M

——— Delaunay Triangulation of M

- - - - Voronoï Diagram of M

One more advantage of the Delaunay triangulation is that it can be computed efficiently. In the 2-dimensional case, Lee and Schacter (LS) have proposed an O(NLogN) algorithm. This algorithm is optimal in the worst-case sense. In the 3-dimensional case, Klee (Kl) placed a lower bound on the worst case complexity of any algorithm computing the Delaunay triangulation by proving that the number of tetrahedra in the Delaunay triangulation of N points is at most $O(N^2)$ (a simple such case will be given below). Edelsbrunner and Seidel shown that the Voronoi diagram and thus the Delaunay triangulation can be computed in that time bound (ES). This is worst-case optimal. However, in most practical situations, the number of tetrahedra is (fortunately) O(N) and a lower bound depending not only on the input but also on the output would be relevant. Practical algorithms have been proposed in the literature to compute the Delaunay triangulation of a set of N points in any dimension. Several authors (Bo, H, W) use an incremental algorithm whose complexity is $O(N^3)$ in the worst case but appear to be almost linear when using fast nearest neighbor search techniques. However, the inherent complexity of the 3-d case make these algorithms much less efficient than their 2-d homologues.

2-2 The points lie on two planar contours

In addition to the general results mentioned above, we now give some specific properties when the points $\mathbb{M}$ lie on two adjacent cutting planes, P_1 and P_2. Let C_1 and C_2 be the contours contained in those planes; both C_1 and C_2 may consist of several connected components. In this section, we will prove that, without any restriction, C_1 and C_2 can be considered to be contained in the Delaunay triangulation of their vertices $\mathbb{M}$ and that this triangulation can be computed using only 2-dimensional operations, thus avoiding the need for a general algorithm computing the Delaunay triangulation in 3-d space.

For i=1,2, let us call M_i the subset of points $\mathbb{M}$ lying in plane P_i, DT_i (resp. V_i) the 2-d Delaunay triangulation (resp. 2-d Voronoi diagram) of points M_i in plane P_i and DT the 3-d Delaunay triangulation of points $M=M_1 \cup M_2$.

Proposition 1: *the intersection of DT with the cutting plane P_i is identical to DT_i (i=1,2).*

Proof: DT is a 3-d simplicial convex complex with all its vertices in P_1 and in P_2. Thus its intersection with P_i (i=1,2) is a 2-d simplicial convex complex. A face of DT $\cap$ P_i is a face of DT_i: this is due to the fact that the intersection of a Delaunay ball with a cutting plane is an empty disk. Conversely, a face of DT_i can always be circumscribed by a sphere whose interior contains no other point of M, provided that its center is sufficiently far from plane P_j, $j \neq i$. Thus each face of DT_i belongs to DT. ♦

An important consequence of Proposition 1 is that the contours C_i will be contained in DT if and only if, in each cutting plane P_i, the contours are contained in DT_i. According to the definition, this condition is satisfied as soon as there exists a set of circles, each of them passing through the vertices of one contour segment and not containing any point in their interior (see Fig.2).

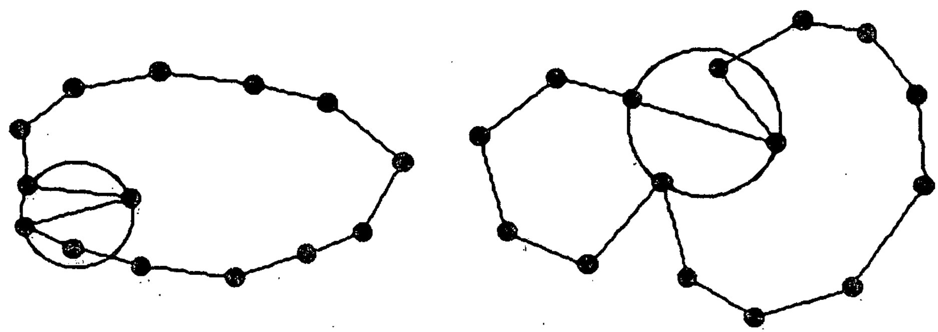


Fig. 2

It is to be noticed that this condition is not very restrictive. In any case, if the condition was not satisfied, we simply need to add some new vertices, as it is stated by the following proposition:

Proposition 2: *every simple polygon is geometrically identical to a polygon contained in the Delaunay triangulation of its vertices, obtained by adding at most a finite number of vertices on some of its edges.*

Proof: Let P be a simple polygon and m a point of one of its edges e. We can associate with m the two maximal balls passing through m and containing only points belonging to e in their interior (see Fig. 3). Because P is simple, the radius of all these balls is greater than $R_0>0$. So only a finite set of them is sufficient to cover the edges of P. In particular, if the set of disks associated to the vertices of P does not cover P, we can add a finite set of points belonging to the edges of P (a new set of vertices) in such a way that the set of disks associated to the old and the new vertices of P covers P. In such a case, P is contained in the Delaunay triangulation of its vertices. ♦

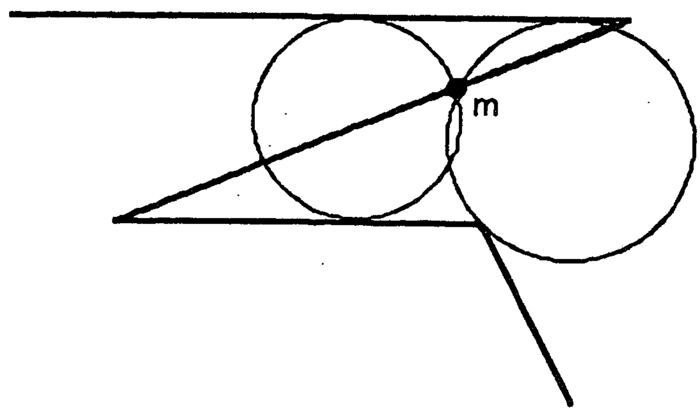


Fig. 3

Thanks to Proposition 2, we will assume now that the contours lying in a cutting plane are contained in the 2-d Delaunay triangulation of the points of M (their vertices) belonging to that plane.

Proposition 1 shows that if we know DT, we know DT_1 and DT_2. We will show now how DT can be computed when DT_1 and DT_2 are known.

We suppose, for clarity, that *the two planes P_1 and P_2 are parallel*. The case where this is not true will be considered in Section 4. The tetrahedra of DT are of two kinds. A tetrahedron T of the first kind (*type T_i*) has three vertices in P_i (i=1 or 2) and consequently, due to proposition 1, these three vertices are the vertices of a triangle t of DT_i. The third vertex of T is the point of M_j ($j \neq i$) which is the nearest from the center of the circumscribed circle of t. The other tetrahedra of DT (*type T_{12}*) do not contain a face in P_1 nor P_2 and consequently, contain exactly one edge in P_1 and one edge in P_2. Due to proposition 1, these edges must be contained in DT_1 and DT_2 respectively. The following proposition reduces the problem of computing such tetrahedra to the problem of finding the intersections of a number of segments in a plane.

Proposition 3: *an edge e_1 of DT_1 and an edge e_2 of DT_2 belong to a tetrahedron of DT iff their dual edges in the corresponding Voronoi diagrams intersect at one point when we project, normally to themselves, P_1 onto P_2.*

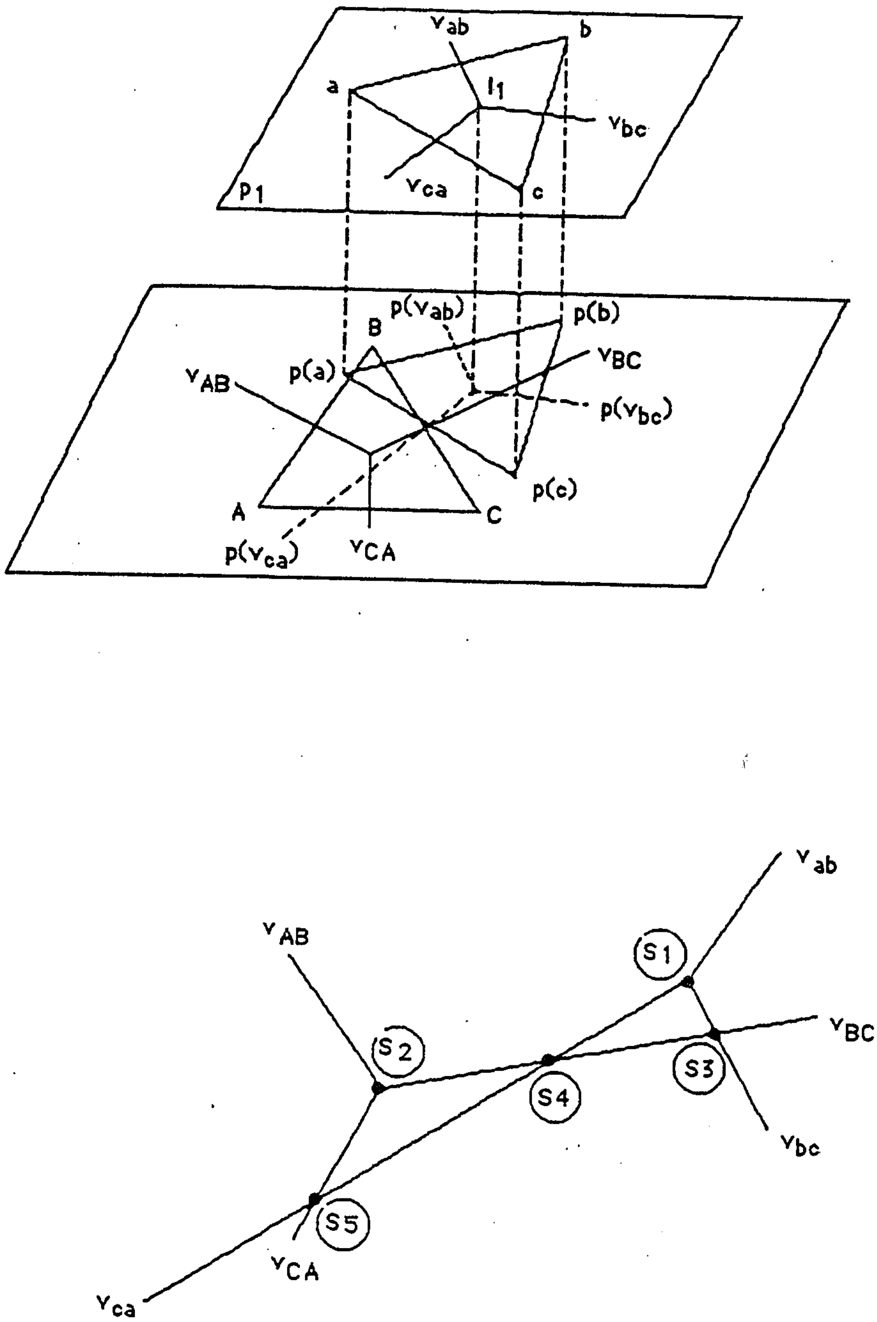
Proof: let e_1=ab and e_2=AB.

1- *the condition is necessary:* for the tetrahedron ($e_1 e_2$) to be contained in the Delaunay triangulation, we need that its circumscribed ball does not contain any point of M_1; so its center must lie in the planar strip ST_1 normal to P_1 and intersecting P_1 along the edge v_1 dual to e_1 in the Voronoi diagram of M_1. For the same reason, it must belong to the planar ST_2 normal to P_2 and intersecting P_2 along the edge v_2 dual to e_2 in the Voronoi diagram of M_2. Thus the center must belong to the intersection of ST_1 and ST_2. This is only possible if ST_1 and ST_2 intersect along a straight line or, equivalently, that v_1 and v_2 intersect at one point when we project, normally to themselves, P_1 onto P_2 (v_1 and v_2 cannot have more than one point in common; otherwise the tetrahedron ($e_1 e_2$) would be flat which is impossible in a Delaunay triangulation provided that (e_1,e_2) is not the only tetrahedron of DT).

2- *the condition is sufficient:* ST_1 and ST_2 intersect along a straight line D. Let O_1 be the intersection of D with P_1 and O_2 the intersection of D with P_2. There exists an unique sphere S whose center O belongs to D, intersecting P_1 along a circle Γ_1 with O_1 as center and $O_1a=O_1b$ as radius and intersecting P_2 along a circle Γ_2 with O_2 as center and $O_2A=O_2B$ as radius. Because O_1 belongs to v_1 the interior of Γ_1 does not contain any point of M; this is also true for Γ_2 and so S is a Delaunay sphere and the tetrahedron ($e_1 e_2$) belongs to DT. ♦

Let us consider the graph $G(M_1,M_2)$ union of the two Voronoi diagrams obtained after projecting P_1 onto P_2 in a direction normal to these planes. A node of this graph is either a vertex of the Voronoi diagram V_i of one of the M_i or an intersection between an edge of the Voronoi diagram V_1 of M_1 with an edge of the Voronoi diagram V_2 of M_2. All the information about the structure of the 3-d Delaunay triangulation of $M_1 \cup M_2$ can be deduced from that new graph. To each node of the graph corresponds a tetrahedron:

1- if the node is a vertex of the Voronoi diagram of M_1(resp. M_2), it is the center of a circle circumscribing a triangle, say abc, of DT_1 (resp. DT_2). When projecting P_1 onto P_2, this node is projected inside a cell of V_2 (resp. V_1) associated to a point of M_2 (resp. M_1), say A. The



v_{ab}
b
a
l_1
v_{bc}
v_{ca}
c
P1
p(v_{ab})
p(b)
B
v_{BC}
v_{AB}
p(a)
p(v_{bc})
p(c)
A
C
p(v_{ca})
v_{CA}
v_{ab}
v_{AB}
S1
v_{BC}
S2
S3
S4
v_{bc}
S5
v_{CA}
v_{ca}

Fig. 4

node corresponds to the Delaunay tetrahedron abcA of type T_1 (resp. T_2). Let us consider a simple example (Fig.4). In both planes P_1 and P_2, the contour consists of a triangle t_1=abc and t_2=ABC. The center of the circumscribed circle of t_i is I_i. The 2-d Delaunay triangulation in each plane P_i consists obviously of the triangle t_i and we call v_{ab}, v_{bc}, v_{ca}, v_{AB}, v_{BC}, v_{CA} the Voronoi edges dual to the edges ab, bc, ca, AB, BC, CA respectively. In this example, we have one tetrahedron of type T_1, S_1=abcB and one tetrahedron of type T_2, S_2=aABC.

2- if the node lying at the intersection between two Voronoi edges v_1 and v_2, dual to two edges e_1 and e_2 of the Delaunay triangulations, it corresponds with the tetrahedron of type T_{12} containing the edges e_1 and e_2. In the example of Fig.4, we have three such tetrahedra: v_{bc} intersects v_{BC} yielding the tetrahedron S_3=bcBC, v_{ca} intersects v_{BC} and v_{CA} yielding the tetrahedra S_4=acBC and S_5=acAC.

The neighboring relationships between the tetrahedra are given by the edges of the graph: for example, S_4 has four neighbours S_1, S_2, S_3 and S_5 while S_1 has two neighbours S_3 and S_4 and two faces on the boundary of the triangulation (which is the convex hull of M).

Let us conclude this section with two remarks which are easy consequences of Proposition 3

Remark 1: DT is independent of the distance between the two planes; a method computing DT by means of Proposition 3 will avoid a number of numerical problems.

Remark 2: the number of tetrahedra of DT may be $O(N^2)$. Indeed consider two line segments, one in P_1 and the other in P_2; let us take N_1 points M_1 on the first one and N_2 points M_2 on the other; V_1 consists of N_1-1 infinite edges and V_2 consists of N_2-1 infinite edges. Each edge of V_i (i=1,2) intersects all the edges of V_j ($j \neq i$). Thus, from Proposition 3, DT will contain $(N_1-1)*(N_2-1)$ tetrahedra (of type T_{12}); see Fig.5. This result is independent of the relative position of the two segments (provided that they are not parallel); in particular they can be as close as we want. But, in that case, the tetrahedra become very flat and their centers go to infinity. So, if we compute only the intersections between the two Voronoi diagrams occuring in a reasonable window containing all the points, we will ignore the very flat tetrahedra (of no interest for our purpose). When the two contours are not too far from one another, this will drastically reduce the number of tetrahedra and so, storage requirement and computing time.

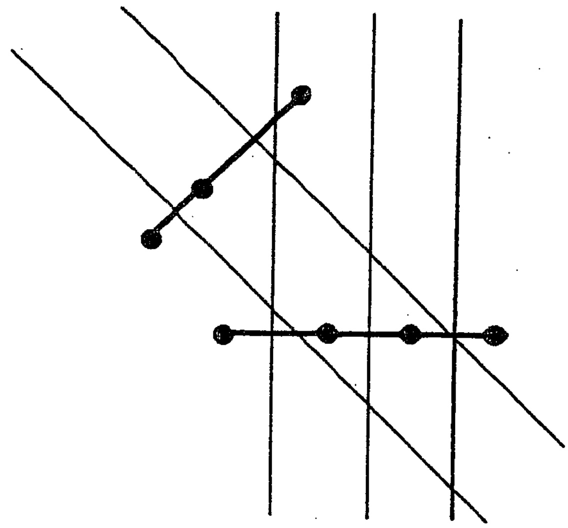


Fig. 5

2-3 An algorithm computing the Delaunay triangulation when the points are distributed on two parallel planes

In the case of points lying on two parallel planes, we can define a specific algorithm, much faster than a general one; this algorithm, which relies heavily on Proposition 3, computes the 3-dimensional triangulation DT from the two 2-dimensional triangulations DT_1 and DT_2 and the graph obtained by projecting plane P_1 onto plane P_2. We note $proj(e_1)$ the projection of edge e_1 of plane P_1 onto plane P_2, normally to P_1 and P_2. Its general structure is the following:

1- Compute DT_1;
2- Compute DT_2;
3- For each triangle t_1 of DT_1, find the vertex m_2 of DT_2 which is closest to the center of the circle circumscribing t_1 and create the tetrahedron (t_1,m_2) of type T_1;
4- For each triangle t_2 of DT_2, find the vertex m_1 of DT_1 which is closest to the center of the circle circumscribing t_2 and create the tetrahedron (t_2,m_1) of type T_2;
5- For each edge v_1 of V_1, find all the edges v_2 of V_2 intersected by $proj(v_1)$, and create the tetrahedron (v_1,v_2) of type T_{12};
6- Create neighborhood relationships between tetrahedra sharing a face.

Let us call N_1 and N_2 the number of points in P_1 and P_2 and $N=N_1+N_2$. As previously mentioned, steps 1 and 2 can be performed in $O(N_1 \mathrm{Log} N_1)$ and $O(N_2 \mathrm{Log} N_2)$ time respectively, which is optimal. Shamos (S), Lee and Schacter (LS) and Guibas and Stolfi (GSt) have proposed such optimal algorithms. Many other algorithms, suboptimal but very efficient in practice, are described in (AEI, GS, SHo, W). Steps 3 and 4 require to find the vertex of a triangulation which is closest to a given point c (the projection of the center of a circumscribing circle), or, equivalently, to find the Voronoi cell in which c lies. This can be done in $O(N_i \log N_i)$ time using $O(N_i)$ storage (i=1,2), which is optimal (Ki). See also, for a general discussion, (EKA). Step 5 and 6 can be solved in O(NLogN+K) time using O(N) storage, if K is the number of intersections (MS, NP). This result has been recently improved by Guibas and Seidel who designed an optimal O(N+K) algorithm (GSe). Thus *the total complexity of the above algorithm is O(NLogN+K)*. We claim that this is optimal. Indeed, O(NLogN) is a lower bound obtained when N_1 (or N_2) is equal to 1: in that case, the 3-dimensional DT is obtained by linking the point in P_1 to the triangles of DT_2. K is the number of tetrahedra of type T_{12} and so is obviously a lower bound. This justifies the claim.

3- SHAPE RECONSTRUCTION

3-1 Statement of the problem

Let us suppose that we have an object O (or a set of disjoint objects which will be also called an object), and that the only information we know about this object is a set of L cross-sections which are the intersections of the object with L planes. *We assume that the sections of the object by the planes each consist of a number of solid regions* :by solid we mean that the boundaries of these regions consist of simple polygons, some possibly lying inside others. The problem is to find a solid object O' whose intersection with the cutting planes is exactly the set of the given cross-sections. We mean by solid a bounded compact set whose boundary is locally everywhere a topological disk, thus preventing regions where the thickness of the object is zero. We reduce the problem of constructing O' over the L cross-sections to one of constructing a sequence of L-1 partial shapes, each of them connecting two cross-sections C_i and C_{i+1} lying on adjacent planes. The method consists, at first, in computing the Delaunay triangulation of the points M_i and M_{i+1} of C_i and C_{i+1}. We have shown, in the previous section (Proposition 2), that the contours bounding the cross-sections can be assumed to be contained in such a triangulation without loss of generality. This will be assumed in the sequel. Moreover the 3-dimensional Delaunay triangulation of the points M_{i-1} and M_i and the 3-dimensional Delaunay triangulation of the points M_i and M_{i+1} intersect the cutting plane P_i along the 2-dimensional DT_i of points M_i: this guarantees the coherence between slice (i-1,i) and slice (i,i+1). So let us consider a slice between two adjacent cutting planes P_1 and P_2. The associated 3-dimensional DT is a convex solid body. To model the portion O_{12} of the object O which lies between P_1 and P_2, we must eliminate a number of tetrahedra. This procedure is repeated for all the pairs of adjacent cross-sections, and, at the end, there remains only a set of tetrahedra O', the reconstructed object, which must satisfy the two conditions previously mentionned:

Condition 1: *O' is a solid.*

Condition 2: *O' intersects the cutting planes along the given cross-sections.*

Before going on, let us make precise what we call a solid in that context.

Definition: *a solid is a set of tetrahedra, connected face-to-face whose boundary is a simple closed polyhedron or an union of disjoint such sets.*

These conditions concern the overall reconstructed object. In order to work slice by slice, we need analogous conditions for a reconstructed slice. Both conditions must be modified. Indeed, consider the two slices (a) and (b) of Fig.6 (dotted regions).

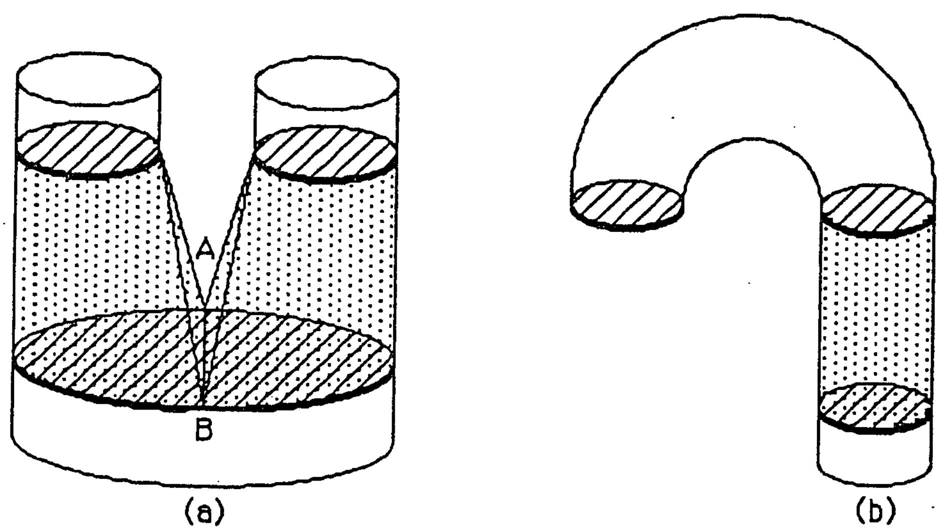


Fig. 6

Though Conditions 1 and 2 are satisfied for the complete objects, they are not satisfied for the slices: in the first example, Condition 1 is not satisfied because the slice has a singularity along edge AB; in the second example, Condition 2 is not satisfied because the intersection of the slice with plane P_1 is strictly included in the cross-section. So we need to relax somehow the above constrained.

Let an *extended slice* ES(S) of a given slice S be the union of S and cylinders, ones lying above the upper plane (say P_1) and intersecting P_1 along the cross-sections of P_1, the others lying below the lower plane P_2 and intersecting plane P_2 along the cross-sections of P_2. The generators of the cylinder are taken to be normal to planes P_i. See Fig.7.

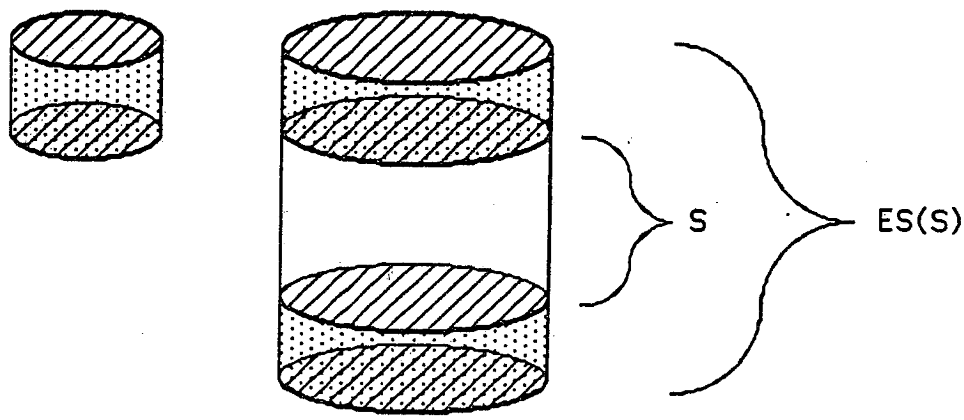


Fig. 7

Definition: *a solid-slice is a slice whose extended slice is solid.*

In order to satisfy conditions 1 and 2, it is necessary to reconstruct solid-slices between each couple of adjacent planes. More precisely, we will extract, from each triangulation between two adjacent planes, a maximal solid-slice. This problem is considered in the following sections 3-3, 3-4 and 3-5. Then we show in section 3-6 that this is also sufficient under weak conditions to reconstruct a 3-d object satisfying Conditions 1 and 2.

3-2 Mark the edges in the cutting planes

Each cutting plane is assumed to have an upper face and a lower face; the upper face of plane P_i is looking at the lower face of adjacent plane P_{i-1}. As a convention, the normals n_i to these planes are oriented in the direction lower-face $\rightarrow$ upper face.

As we assumed at the begining, the contours are oriented in order to separate the inside of the object from the outside. Suppose the contours are oriented in clockwise order around the normals of the corresponding planes. It is then easy to know if an edge M_iM_j is inside, outside or on the contour. M_iM_j is inside if M_iM_{i-1}, M_iM_j, M_iM_{i+1} are in counterclockwise order, M_iM_j is outside if M_iM_{i-1}, M_iM_j, M_iM_{i+1} are in clockwise order, M_iM_j is on the contour if $M_j=M_{i-1}$ or $M_j=M_{i+1}$.

3-3 Find the inside tetrahedra between two cross sections

We now consider two adjacent cutting planes P_1 and P_2, and the Delaunay triangulation DT of the corresponding points $M=M_1 \cup M_2$.

Remember that all the information about DT is contained in the graph $G(M_1,M_2)$, noted G for short in the sequel, defined in Section 2-2. This graph is obtained by superimposing the two planar Voronoi diagrams V_1 of points M_1 and V_2 of points M_2. Each node represents a tetrahedron of DT and two nodes are adjacent iff they represent tetrahedra with a face in common. Three other easy properties of that graph are the following:

Property 1: *the set of all the tetrahedra of DT sharing a common edge e in P_i (i=1 or 2) are represented by all the nodes of G belonging to the edge of V_i dual to e.*

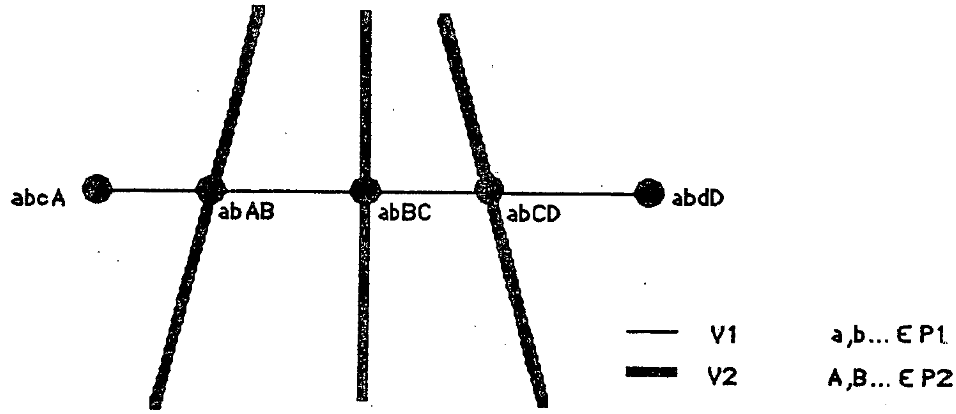


Fig. 8

Fig. 8 shows an example. Edge ab belongs to two tetrahedra of type T_1, abcA and abdD, and to three tetrahedra of type T_{12}, abAB, abBC, abCD. We see that an edge e of V_i intersected by k edges of V_j ($j \neq i$) gives rise to k+1 edges in G: these k+1 edges will be said to be *part of e*.

Property 2: *the set of all the tetrahedra of DT sharing a common vertex m of P_i ($i=1,2$) are represented by all the nodes n(m) of G belonging to the boundary and to the inside of the cell of V_i surrounding m. We note G(m) the subgraph of G whose vertices are the n(m).*

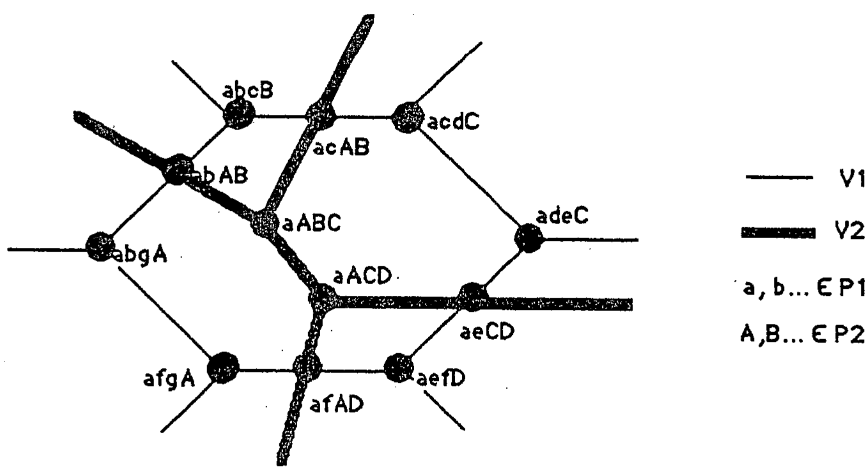


Fig. 9

Fig. 9 shows an example: point a belongs to 6 tetrahedra of type T_1 (abgA, abcB, acdC, adeC, aefD, afgA), 4 tetrahedra of type T_{12} (abAB, acAB,aeCD,afAD) and two tetrahedra of type T_2 (aABC, aACD).

Property 3: *the set of all the tetrahedra of DT sharing a common edge m_1m_2 with m_1 in P_1 and m_2 in P_2, are represented by all the nodes of G belonging to an elementary cycle of G. We note $G(m_1, m_2)$ such a cycle.*

In a way similar to that of section 2-2, we speak of a node of type T_1, T_2 or T_{12} according as it represents a tetrahedron of type T_1 (i.e., with a face in P_1), T_2 (i.e., with a face in P_2), or T_{12} (i.e., with an edge in P_1 and an edge in P_2).

Which tetrahedra in DT belong to the model S_{12} of the slice O_{12} of the object O? We will describe the necessary operations from both a geometrical and a graph-theoretical points of view. This latter will prove to be more powerful in providing a good algorithm and proofs of correctness. However, in order to assist the reader's intuition, we give also a geometrical description.

If a tetrahedron of type T_1, T_2 or T_{12} contains an edge in plane P_1 or in plane P_2 which is external to the given contours, then it does not belong in the model S_{12}. Otherwise, Condition 2 of section 3-1 would not be satisfied. In the graph G, that means that we delete the nodes representing the tetrahedra containing the edges which are external to the contours and the edges adjacent to those nodes in G yielding G_1 (*Rule 1*). Once this rule has been applied, it remains to satisfy condition 1. The correct representation of the slice will show connected components of one layer (or 2-dimensional pieces of these components) solidly connected to similar pieces of connected components in the other. How do we achieve this solidity of connection? We must avoid the situation where, even locally, a portion of the triangulation D_1 is connected only to a k-dimensional (k=0 or 1) portion of the triangulation D_2. To have such a weak connection would produce a model with extraneous veils and linkages yielding a non-solid reconstructed slice. Fortunately, we can check this behaviour at each edge e internal to or on the boundary of one of the given contours. In the triangulation DT, such an edge e (say in plane P_2) is contained in a sequence S_e of tetrahedra, joined face-to-face, beginning with a tetrahedron of type T_2, then one or more tetrahedra of type T_{12}[2]. If the edge e is internal to a contour in P_2 then this sequence will end in a single tetrahedron of type T_2, otherwise it ends in an intermediate tetrahedron of type T_{12} having a face on the boundary of DT. To obtain a solid representation of the slice O_{12}, we must be sure that the tetrahedra which remain attached to such an edge e in S_{12} are still attached face-to-face to at least one of the two possible tetrahedra of type T_2. Otherwise, the local region of contact of this portion of the model with plane P_2 will be one-dimensional, not solid. In this way, we arrive at a

---

2: Note that a tetrahedron of type T1 cannot share a common face with a tetrahedron of type T2.

criterion describing intermediate tetrahedra which must be eliminated. For any tetrahedron S of type T_{12} which remains after the exclusion of tetrahedra with edges external to C_1 or C_2 (rule 1), we perform the following test. Say the tetrahedron S has edges e in plane P_1 and f in plane P_2. These edges are either interior or boundary edges. We verify that S is connected face-to-face in the sequence S_e, through tetrahedra which remain in the so-far-sculpted triangulation, to at least one of the two possible tetrahedra of type T_1, and is also connected in the same way in the sequence S_f to at least one of the two tetrahedra of type T_2. If this is true, we say that S belongs to a *solid path*. If not, then the tetrahedron S is rejected *(rule 2)*.

The above solidity test can also be done on graph G_1. Let n be a node of type T_{12} of G_1 lying at the intersection of an edge v_1 of V_1 and an edge v_2 of V_2, and let S be the corresponding tetrahedron. S belongs to a solid path iff n is connected, through at least one sequence of edges being part of v_1, to a node of type T_1 and is connected, through at least one sequence of edges being part of v_2, to a node of type T_2.

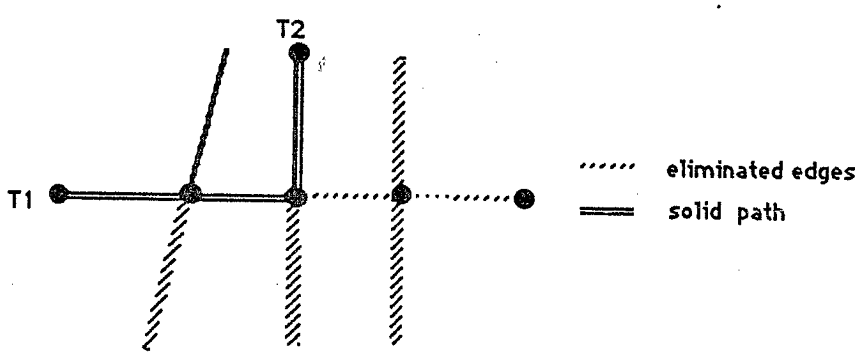


Fig. 10

When a node has been eliminated during the process, it cannot be an element of a solid path of any other node; this property can be seen in G_1: we orient the edges of G_1 which are part of v_1 or v_2, from n towards the nodes of type T_1 or T_2 (see Figure 11). In order to deal with the infinite edges of G_1, we assume that each infinite edge ends with a node at infinity. This node does not correspond to a tetrahedron of the reconstructed object and so, must be eliminated. The infinite edges are then oriented in the same way as the finite edges.

A node n of type T_{12} belongs to a solid path iff at least two edges of G_1 come out from n, one of them being part of an edge of V_1, and another one being part of an edge of V_2. When an edge of G_1 has received both orientations, it cannot be used in a solid path and must be eliminated.

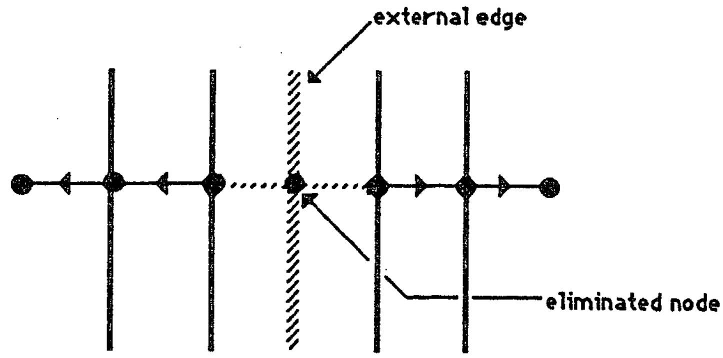


Fig. 11

It remains to decide which tetrahedra of type T_1 or T_2 are to be retained. Let us consider a set of tetrahedra with the same type T_i (i=1 or 2), retained after the application of rule 1, and intersecting plane P_j ($j \neq i$) at the same point. These tetrahedra are connected face-to-face. If such a set has an element which shares a face with a retained tetrahedron of type T_{12}, all the set will be retained (*rule 3*). It can be easily checked that this rule does not create any singularity along an edge and that no other tetrahedron can be retained without creating a singularity at a vertex in P_1 or P_2.

In graph G_1, this rule can be easily applied after an appropriate contraction of the graph. If two nodes with the same type T_1 or T_2 are adjacent, we consider them as a single node of type T_1 or T_2. A contracted node (and thus all the corresponding nodes of G_1) is eliminated iff it has no adjacent edge. From the above discussion, we deduce the following procedure:

Procedure slice

1- put the infinite nodes and the nodes of type T_1 or T_2 with an external face in a list of "eliminated nodes";

2- for each edge v of $V_1 \cup V_2$ dual to an external edge **do**

  21- eliminate from G the edges part of v;

22- eliminate from G the nodes belonging to v and add them to the list of "eliminated nodes";

3- for each eliminated node n of type T_{12} do (* $n=v_1 \cap v_2$ *)

31- eliminate from G the (non yet eliminated) edges adjacent to n;

32- starting from n, orient the edges part of v_1 and the edges part of v_2 until you find either an edge with the same orientation or or a vertex of type T_2. If an edge $e=(n_a, n_b)$ with the opposite orientation, say $n_a \rightarrow n_b$, is traversed, eliminate e and node n_b, and add n_b to the list of "eliminated nodes";

4- eliminate each contracted node of type T_i without any adjacent edge.

3-4 Complexity analysis

Step 1 requires O(N) time and Step 2 requires O(E) time if E is the number of edges of G. Clearly E=O(T) if T is the number of tetrahedra of DT.

Let e be an edge of G. We assume, without loss of generality, that e is part of an edge of V_1. During step 3, three cases may occur:

1- e is eliminated and will not be considered any longer;

2- e receives an orientation; this case can only occur twice since e cannot receive more than two (opposite) orientations;

3- e is found to have the same orientation as the propagated orientation. Notice that e is the first edge encounted by the propagation. This case can only occur once since the edge adjacent to e in the direction opposite to the orientation of e has now received the same orientation as e. Thus e cannot be the first edge encounted by a future propagation with the same orientation.

Thus, during Step 3, e cannot be considered more than three times and so, the complexity of steps 1, 2 and 3 is O(E)=O(T). The complexity of step 4 is obviously O(T), thus *the overall complexity of procedure SLICE is O(T).*

3-5 Correctness of procedure SLICE

Procedure SLICE has been proposed in order to satisfy some necessary conditions. This section will establish under what conditions the algorithm effectively produces an extended-solid slice. We consider successively the two possible cases of singularity:

singularities along edges and singularities at vertices. We note G_{12} the graph produced by the algorithm and S_{12} the associated set of tetrahedra (*reconstructed slice*).

We give some preliminary definitions and some classical results from algebraic topology. We call *shell of an edge e* in a complex the set of all the edges f such that (e,f) is a tetrahedron of the complex and such that e and f are disjoint. No singularity occurs along an edge iff its shell is either a cycle (if the edge is interior to the complex), or an arc (if the edge lies on the boundary of the complex). In a similar way, we call *shell of a vertex m* in a complex the set of all faces f such that (m,f) is a tetrahedron of the complex and such that m does not belong to f. No singularity occurs at a vertex iff its shell is either a topological sphere (if the point is interior to the complex), or a topological disk (if the point lies on the boundary of the complex).

Proposition 4: *procedure SLICE does not produce any singularity along edges lying in one of the cutting planes.*

Proof: we start with an easy lemma.

Lemma: *let v be an edge of V_1 or V_2, and e and e' two edges of G, part of v. If e and e' have been eliminated and, thus, are not in G_{12}, then all the edges part of v lying between e and e' cannot belong to G_{12}.* (Indeed, these edges have received the two opposite orientations, one when considering e, the other when considering e', and so, have been eliminated).

Proof of Proposition 4: Let us consider an edge e of S_{12} lying in one of the cutting planes, say P_1. e lies on the boundary of DT. We call R_{ij} the jth contour of plane P_i. We note shell(e)=...e_{2i}-E-e_{2j}... if the shell of e in DT contains a sequence e_{2i} of edges interior to or on the boundary of contour R_{2i}, then a sequence E of edges belonging to some eliminated tetrahedra (not in S_{12}), then a sequence e_{2j} of edges interior to or on the boundary of contour R_{2j}. Moreover we note e_{12} an edge with one end-point in P_1 and the other in P_2. The contours are supposed to consist of disjoint simple polygons, so the shell of e cannot contain the sequence ...e_{2i}-e_{2j}... ($i \neq j$). Moreover, from the lemma, it cannot contain the sequence ...E-e_{2i}-E... nor end with the sequence ...E-e_{2i} (in case where e belongs to the convex hull of M_1). Thus the possible cases of shell of an edge belonging to one of the cutting planes are:

a/ e_{12}-e_{2i}-e_{12} or e_{12}-e_{2i}

b/ e_{12}-e_{2i}-E-e_{12} or e_{12}-e_{2i}-E

c/ e_{12}-e_{2i}-E-e_{2j}-e_{12}.

In cases a/ and b/, the shell of e in S_{12} is an arc; so there is no singularity along edge e. This remains true if we extend S_{12} as described in Section 3-1. Case c/ yields a singularity along edge e but, if we extend S_{12} as described in Section 3-1, the two arcs e_{2i} and e_{2j} are linked by an arc lying inside the extension. So the singularity disappears. ♦

For convenience, we add the point at infinity: in graph G, we add, on each infinite edge, a node (the infinite node) and we also add the edges representing the adjacency relationships between the infinite tetrahedra. Thus each cell of G is closed. These additional nodes cannot represent tetrahedra of the reconstructed slice and so cannot belong to G_{12}.

Proposition 5: *Let $m_1 m_2$ be an edge of S_{12}, with m_1 in P_1 and m_2 in P_2. S_{12} has no singularity along $m_1 m_2$ iff $G_{12}(m_1 m_2)$ is connected.*

Proof: the shell of $m_1 m_2$ is an arc iff $G_{12}(m_1 m_2)$ is connected. ♦

Proposition 6: *Let m be a vertex of M. The extended reconstructed slice ES_{12} associated to S_{12} has no singularity at m iff $G(m)$-$G_{12}(m)$ is connected.*

Proof: geometrically the proposition says that ES_{12} has no singularity at m iff the set SE of the eliminated tetrahedra containing m is connected face-to-face. Our proof uses a geometrica argument. Let EDT be DT extended below the lower plane and above the upper plane as previously described in section 3-1; thus EDT=DT $\cup$ (ES_{12}-S_{12}). DT is convex and solid; the contours being simple polygons, EDT is also solid and the shell of a vertex of EDT is a topological sphere.

The condition is *necessary*: we prove the converse. If SE is not face-to-face connected, then th shell of m in SE is made of at least two disks, disjoint or possibly touching at a vertex. The shell of m in ES_{12} is equal to the difference between the shell of m in EDT and the shell of m in SE. Consequently the shell of m in ES_{12} is a sphere with at least two holes (possibly touching at a vertex), not a disk. So ES_{12} has a singularity at m.

The condition is *sufficient*: Let us suppose that SE is face-to-face connected. Due to Proposition 5, S_{12} has no singularity along any edge joining m to a vertex of the plane not containing m. Thus the shell of m in SE is a disk and the shell of m in the extended reconstructed slice ES_{12} is a disk, which means that ES_{12} has no singularity at m. ♦

It is to be noticed that Proposition 6 is stronger than Proposition 5: if no singularity occurs at any vertex, then no singularity can occur along an edge. In such a case, procedure SLICE will produce the *maximal* solid-slice contained in DT. However the reconstructed slice produced by procedure SLICE may not satisfy the condition of Proposition 6. Roughly speaking, such a situation occurs when the two cross-sections are too different: no good solution, with respect to Conditions 1 and 2 of Section 3-1, is contained in DT and so can be obtained by the procedure. We depict in Fig.12-14 three typical cases.

If no other measurement can be performed, the natural solution to overcome the difficulty is to interpolate new cross-sections. We create one or two intermediate cross-sections between P_1 and P_2 not too different from C_1 and C_2 thus allowing to create (two or three instead of one) solid-slices. The union of these two or three slices will be an extended solid-slice. With a different motivation, interpolations of new contours have been proposed by Sloan and Hrechanyk (SH): they use shape descriptors to hypothesize a transformation which accounts for the difference in shape between successive contours. Our construction does not require to make any hypothesis: we simply add to G_{12} the nodes creating the singularities. The reconstructed slice S_{12} thus obtained will be an extended solid slice but with possibly some additional edges and/or faces which are known to be exterior to the contours. Suppose that the intersection of S_{12} with, for instance, P_1 has additional parts; these additional parts define an intermediate cross-section C'_1 which contains the actual one C_1. We place C'_1 between P_1. and P_2 and create the slice between C_1 and C'_1 by simply computing the cylinder between the two identical contours, one in C_1 and the other in C'_1. Thus we allow the creation of additional "horizontal" (i.e. parallel to the cutting planes) faces; remember that we needed procedure SLICE to create "horizontal" faces in case of a new contour (cf Fig.4b).

Procedure CORRECT

for each point m of M_i (i=1,2) **do**

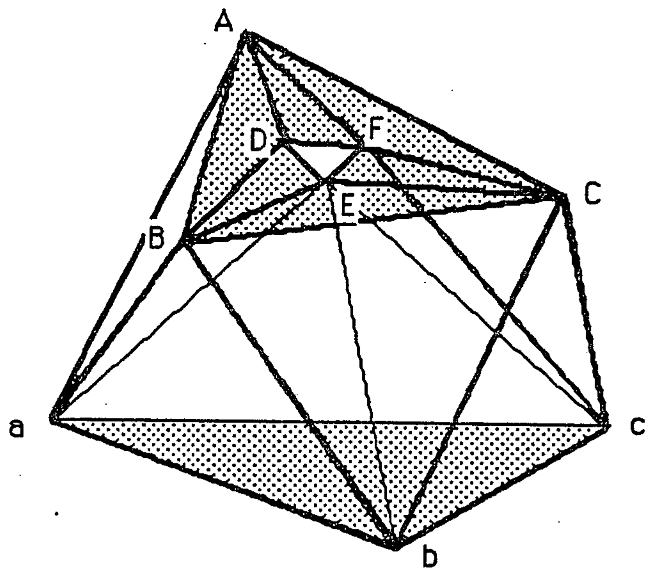
  for each connected component $\mathcal{C}$ of $G(m)-G_{12}(m)$ **do**

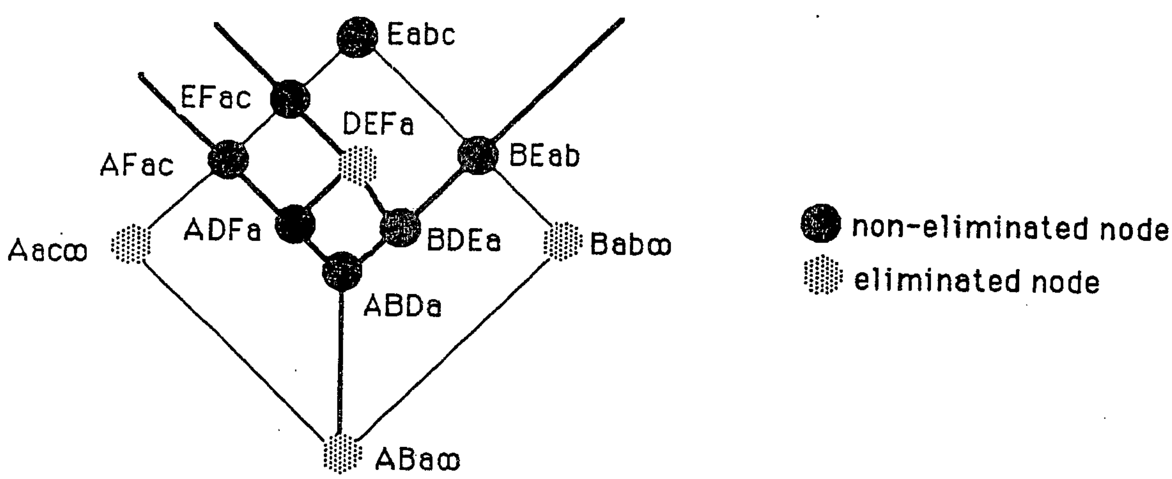
  if $\mathcal{C}$ does not contain the infinite node **then**

   1- add all the nodes of $\mathcal{C}$ to G_{12}: this create a new cross-section C'_i we place between planes P_1 and P_2, closer to P_i than to P_j;

   2- compute the (trivial) slice between C'_i and C_i
   (* this can be done by computing the (trivial) DT of C_i and C'_i and applying step 1 of procedure SLICE *)

The complexity of procedure CORRECT is clearly O(T). We show in Fig.15-17 the slices produced by procedure CORRECT, corresponding to Fig.10-12.


A
D
F
E
C
B
a
c
b


Eabc
EFac
DEFa
AFac
BEab
Aacoo
ADFa
BDEa
Baboo
ABDa
ABaoo
non-eliminated node
eliminated node

G(a)

Fig. 12

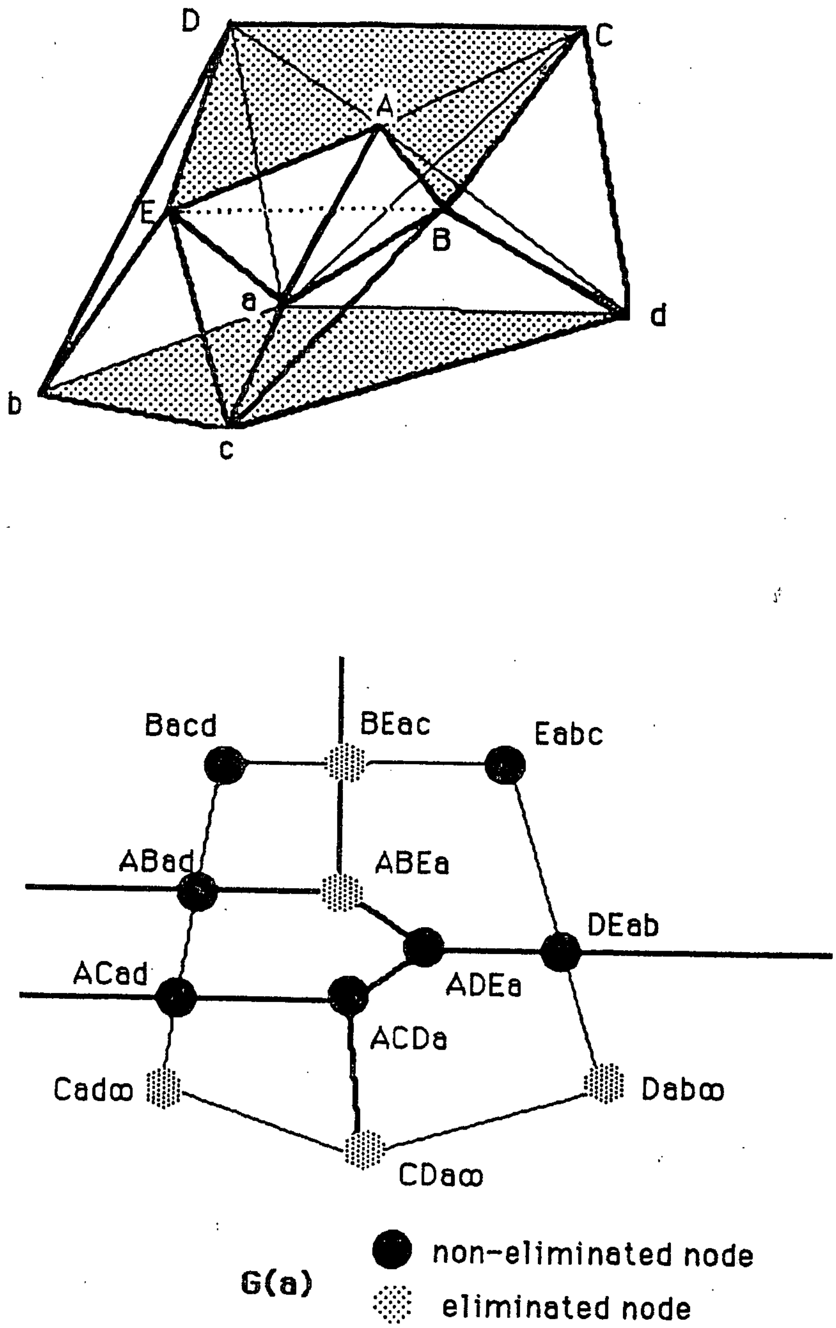
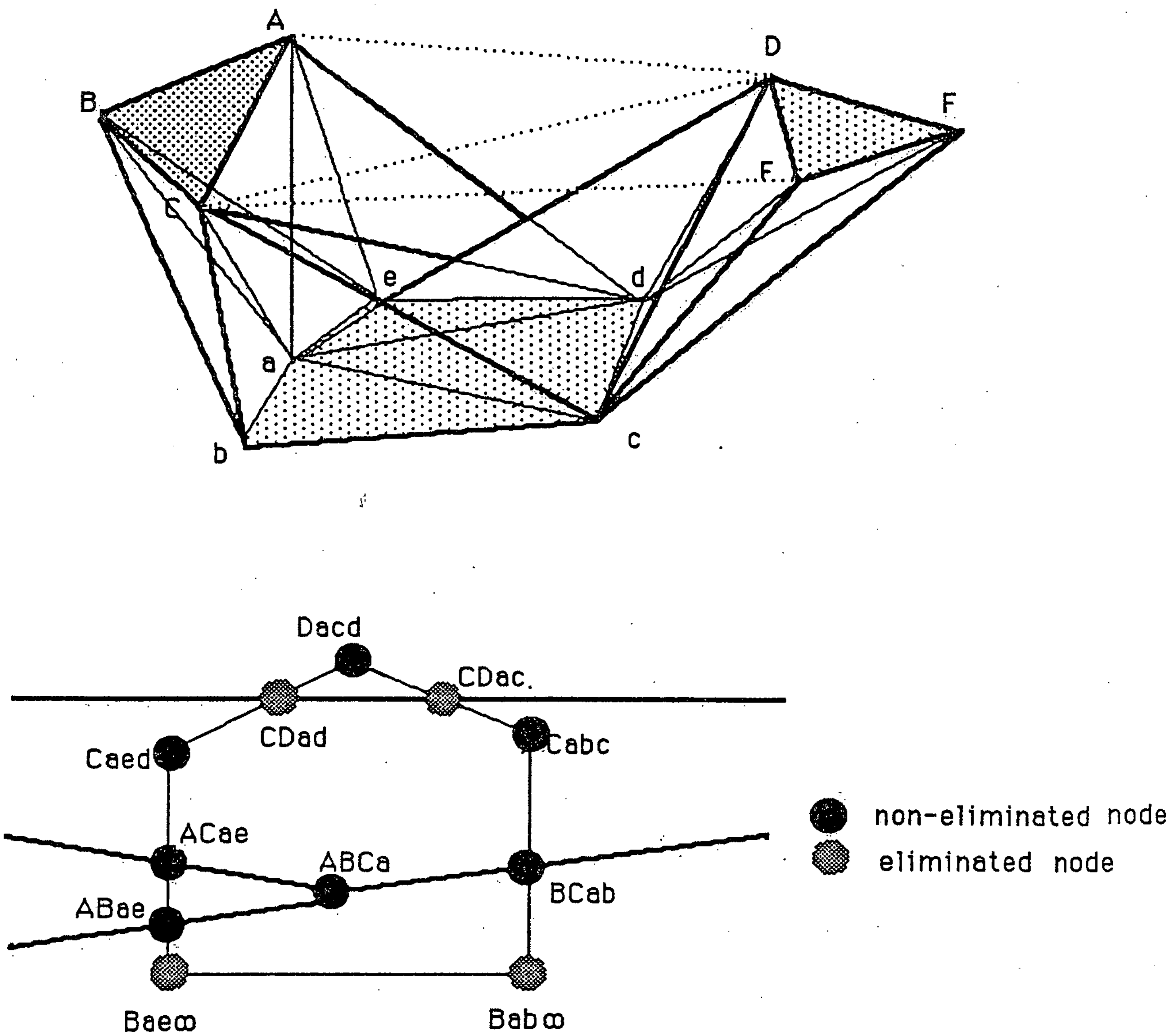

D
C
A
E
B
a
d
b
c
Bacd
BEac
Eabc
ABad
ABEa
DEab
ACad
ADEa
ACDa
Cadœ
Dabœ
CDaœ
G(a)
non-eliminated node
eliminated node

Fig. 13


A
D
B
F
C
F
e
d
a
b
c
Dacd
CDac.
CDad
Caed
Cabc
ACae
ABCa
ABae
BCab
Baeω
Babω
non-eliminated node
eliminated node

G(a)

Fig. 14

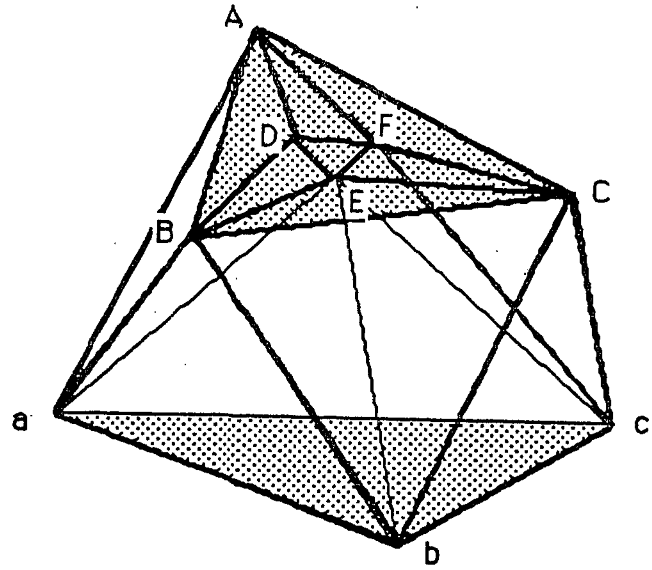


Fig. 13

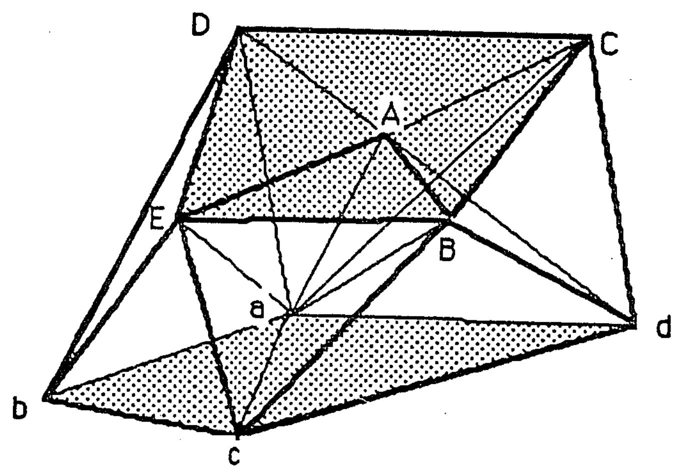


Fig. 14

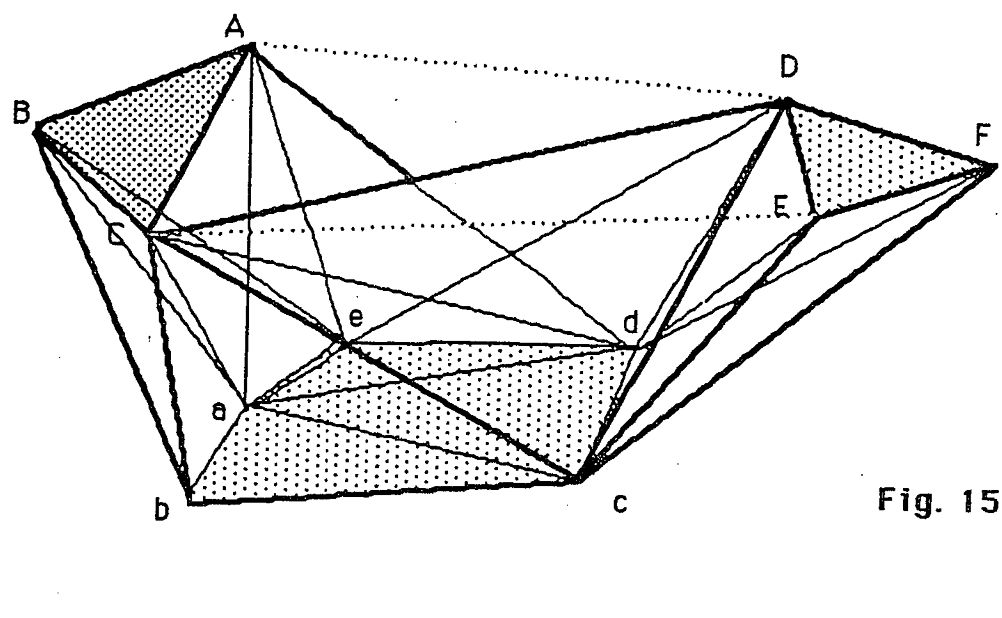
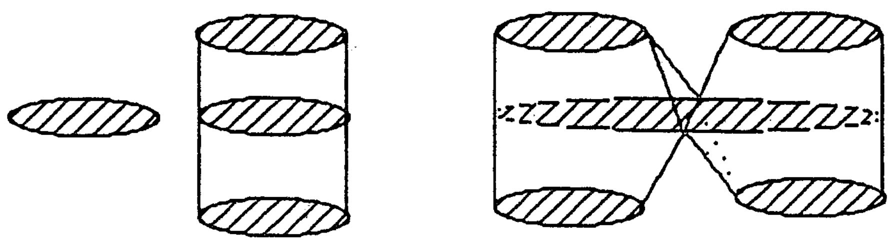


Fig. 15

3-6 Overall reconstruction

In order to reconstruct the overall object, we apply the above method to the L-1 slices. The overall reconstructed object will, in most situations, satisfy Conditions 1 and 2 of section 3-1. However Condition 2 may be violated if a face of a cutting plane P_i is found to be horizontal in the two adjacent slices, $(P_{i-1}P_i)$ and (P_iP_{i+1}). In that case (see Fig. 18a), the reconstructed object will intersect P_i along only a subset of the given cross-sections. Condition 1 may also be violated if an edge of a cutting plane P_i is found to have a singularity (case c/ of Proposition 4) in the two adjacent slices containing the edge (see Fig. 18b).



(a) (b)

Fig. 18

These difficulties occur when the distance between the cutting planes is too large; in order to avoid these cases, and to prevent the creation of non-solid objects, we need to perform a new intermediate cut of the object or, if this is not possible, to interpolate new cross-sections: we duplicate cross-section C_i of plane P_i, yielding a new cross-section C'_i, we place close to plane P_i. The slice between C_i and C'_i can be computed in a trivial way and will guarantee that no face of a cross-section is lost and that no singularity occurs.

4- FINAL REMARKS AND EXPERIMENTAL RESULTS

1- It is to be noticed that the algorithm produces one of the correct solids contained in the Delaunay triangulation, namely the maximal one. Other correct solutions, satisfying Conditions

1 and 2 of Section 3-1, can be obtained by eliminating all or some of the tetrahedra with an edge on a contour in one cutting plane, and an edge on a contour in the other cutting plane. This can be done by eliminating one such tetrahedron at a time, provided that it is adjacent to exactly two non eliminated tetrahedra (B2). This allows a variety of possible options for choosing the final shape, which may depend on the application and on the a priori knowledge we have about the object.

2- In our solution, we accept "horizontal" faces (i.e. belonging to one of the cutting planes). This is to be compared to the solutions proposed in (K),(FKU) and their variants (GD),(SH) where no horizontal faces are allowed. Such an a priori constraint may yield unsatisfactory results in cases where there exist horizontal or close to horizontal faces. See Fig.19.

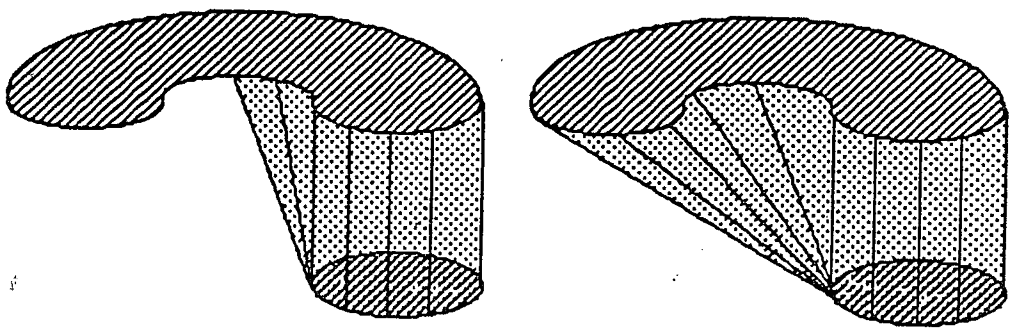


Fig. 19

3- The method can be used even if the planes are not parallel. The algorithm presented in Section 3 cannot be used any longer but can be slightly modified in order to produce some other triangulations. For example, instead of the usual DT, we can compute a curved DT obtained by rotating the planes P_1 and P_2 around their common axis $P_1 \cap P_2$, instead of projecting P_1 onto P_2. The structure of this new triangulation is also a planar graph and the elimination process remains unchanged.

4-The geometric dual of any triangulation of a set of points distributed in two planes is a planar subdivision whose skeleton is a graph with Properties 1, 2 and 3 of Section 3-3. Thus the reconstruction algorithm can make use of any kind of triangulation. The usual Delaunay triangulation, based on the Euclidean distance, seems the most appropriate if no a priori knowledge on the shape is available: indeed, it relates each point to its nearest neighbors in the different directions (a precise result can be found in (B2)). However, in some applications, other triangulations and, among others, Delaunay triangulations based on other metrics may be useful.

5- An implementation of this algorithm has been written in C. Fig. 20 shows a set of contours The partial surfaces are presented in Fig. 21; for clarity, only the faces of the boundary of the non eliminated tetrahedra not lying in a cutting plane are shown. Fig. 22 contains a shaded image of the resulting overall surface. Other results on medical images are shown in Fig. 23 and 24.

5- CONCLUSION

We have presented a new method to reconstruct a volumetric model of an object or of a set of polyhedral objects known by a finite number L of planar cross-sections. The method reduces the problem of constructing a shape over the L cross-sections to one of constructing a sequence of L-1 partial shapes, each of them connecting two cross-sections lying on adjacent planes. The solution makes use of the Delaunay triangulation which is isomorphic, in that specific situation where the points lie in two planes, to a planar graph, obtained by superimposing two planar Voronoi diagrams. This graph, which may be of interest in its own right, permits us to compute the 3-dimensional Delaunay triangulation of points lying in two planes in an optimal way by means of 2-dimensional operations only: the complexity of our algorithm is O(NLogN+T) where N is the size of the input (the number of points) and T the size of the output (the number of tetrahedra of the triangulation).

We compute the L-1 Delaunay triangulations of the vertices of the L-1 couples of adjacent cross-sections. Shape reconstruction is then achieved, slice by slice, by pruning these L-1 Delaunay triangulations. The method can handle the case of multiple contours in the cutting planes. The number of contours may be different from one cross-section to the other, with some possibly included in others. Moreover the method guarantees that the result is a solid object. This object is represented by a set of tetrahedra which makes easy many calculations. for instance the calculus of the volume of the objects, or the production of meshes for finite element methods. The complexity of the reconstruction algorithm is O(T) if T is the maximal number of tetrahedra in the Delaunay triangulation between two adjacent planes P_i and P_j, containing $N=N_i+N_j$ points. T is $O(N^2)$ in the worst-case but, when the contours in two adjacent cross-sections are not too different, it is expected to be O(N).

Finally, we mention two possible applications of the work reported here. At first , we note that a set of (polygonal) cross-sections of an object is a very compact representation of the object. As soon as a triangulation and a set of cross-sections satisfying our hypothesis (cf. Proposition 2 and 6) have been chosen, this representation is equivalent to a volumetric representation of the object, which can be obtained by the above method. Another possible application is related to shape-matching: the shape reconstruction problem can be considered as a special case of a more general shape-matching problem. Similar problems appear in other domains, most notably that of motion perception. Both the problem of recognizing the motion of a set of objects through space (and time) and the problem of reproducing such a motion (as in key frame animation) can be usefully viewed as shape matching processes.

ACNOWLEGMENTS

The author is grateful to Henry Crapo for helpful discussions about the solidity test and to Bernhard Geiger for his contribution to the implementation of the algorithm.

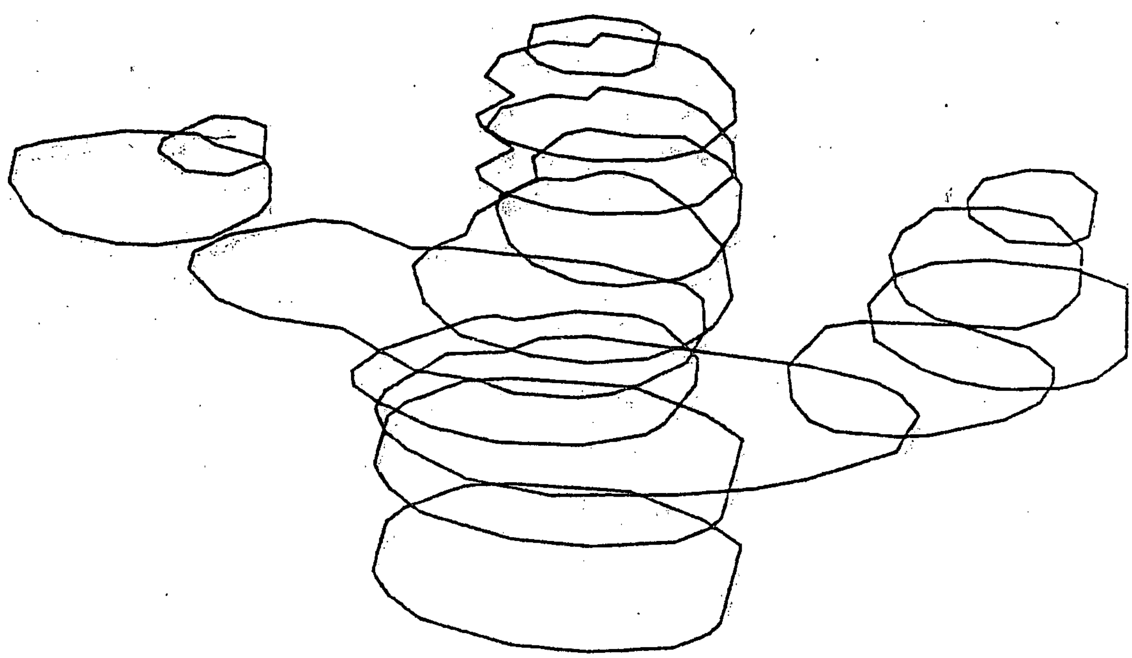


Fig. 20

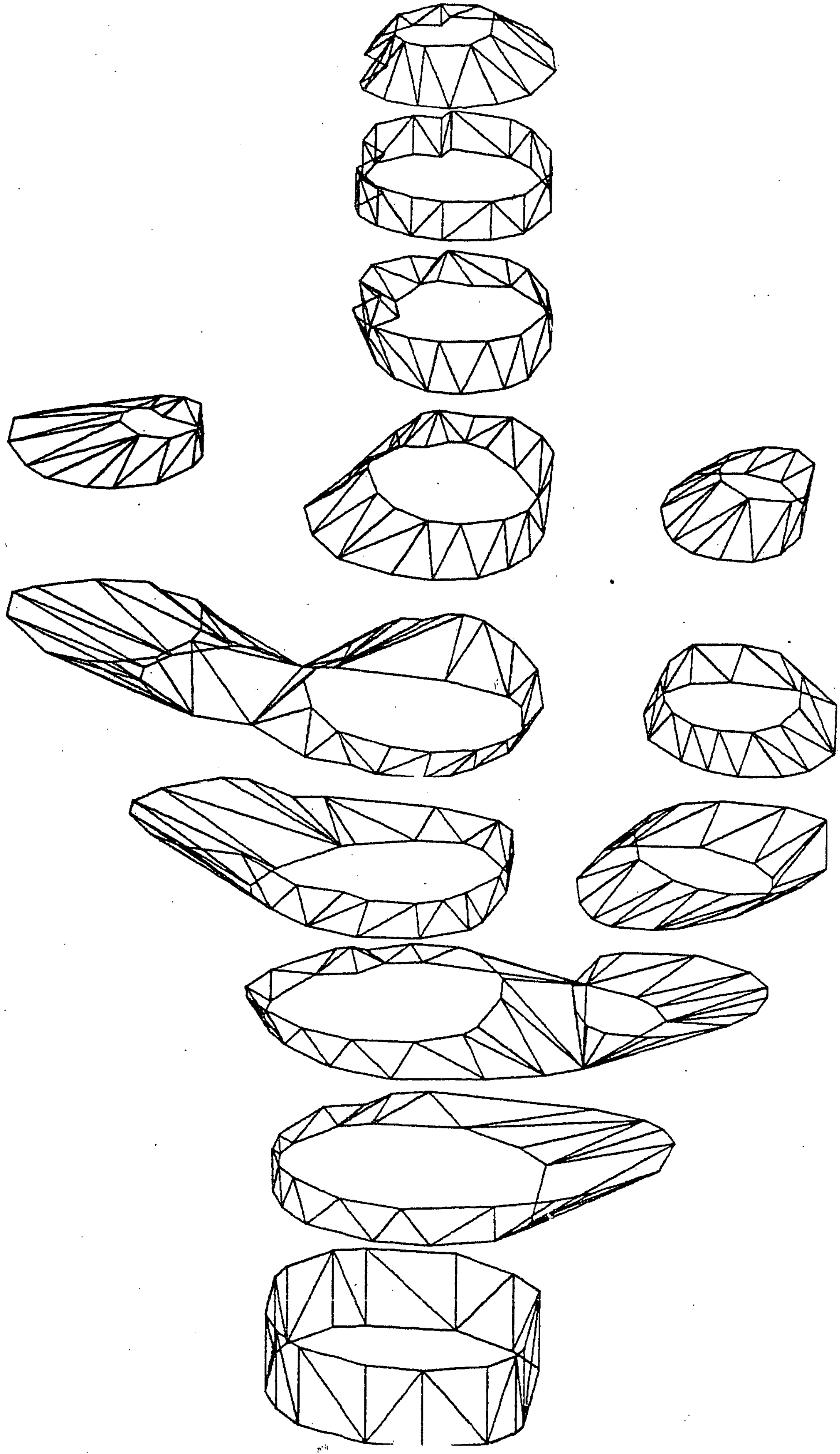


Fig. 21

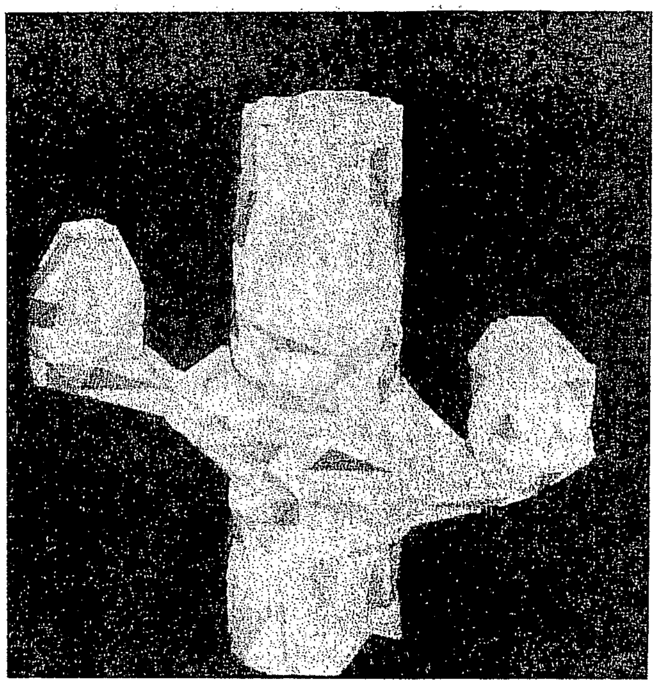


Fig. 22

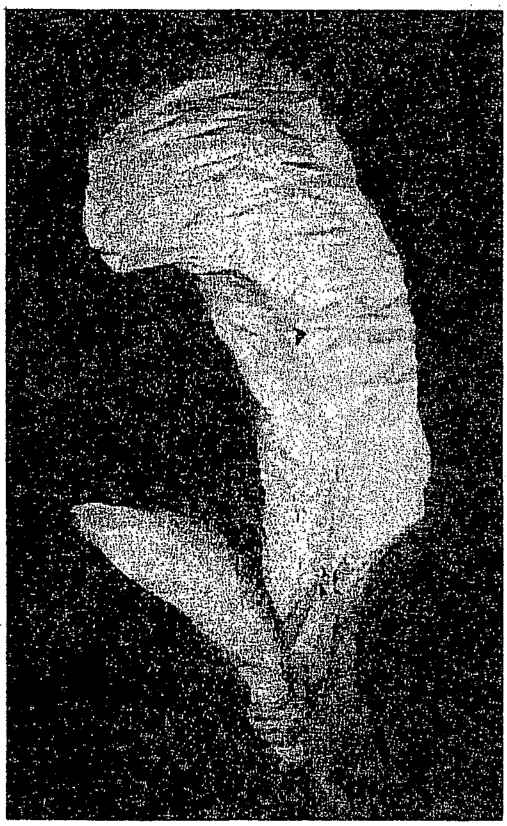


Fig. 23

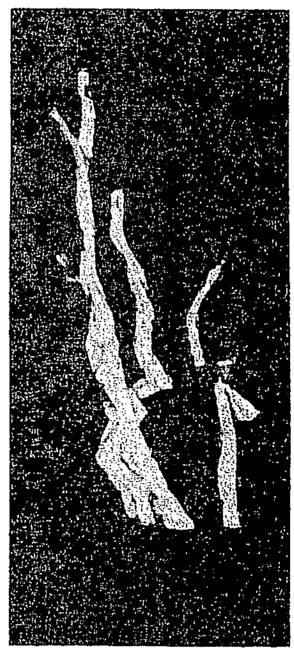


Fig. 24

REFERENCES

(AFH) **Artzy, E., Frieder, G., Herman, G.T.**, The Theory, Design, Implementation and Evaluation of a Three-Dimensional Surface Detection Algorithm, Computer Graphics and Image Processing, Vol. 15, 1981, pp. 1-24.

(AEI) **Asano, T., Edahiro, M., Imai, H., Iri, M.**, Practical Use of Bucketing Techniques in Computational Geometry, in Toussaint, G.T. (ed.), Computational Geometry (North Holland 1985), pp. 153-195.

(B1) **Boissonnat, J.D.**, Shape Reconstruction from Planar Cross-Sections, Proceedings of the IEEE conf. on Computer Vision and Pattern Recognition, San Francisco, June 1985, pp. 393-397.

(B2) **Boissonnat, J.D.**, Geometric structures for 3-dimensional shape representation, ACM Trans. on Graphics, October 1984, pp. 266-286.

(Bo) **Bowyer, A.**, Computing Dirichlet tesselations, Computer Journal, Vol. 24, N° 2, 1981, pp. 162-166.

(EKA) **Edahiro, M., Kokubo, I., Asano, T.**, A New Point-Location Algorithm and its Practical Efficiency - Comparison with Existing Algorithms, ACM Trans. on Graphics,Vol.3, N° 2, April 1984, pp. 86-109.

(ES) **Edelsbrunner, H., Seidel, R.**, Voronoi Diagrams and Arrangements, Proceedings of the ACM, Symp. on Computational Geometry, Baltimore, June 1985, pp. 251-262. Also in Discrete and Computational Geometry, Vol. 1, N° 1, 1986, pp. 25-44.

(FKU) **Fuchs, H., Kedem, Z., Uselton, S.P.**, Optimal surface reconstruction from planar contours, Comm. of the ACM, Vol. 20, Num. 10, October 1977, pp 693-702.

(GD) **Ganapathy, S., Dennehy, T.G.**, A New Triangulation Method for Planar Contours,, ACM Computer Grahics, Vol. 16, July 1982, pp. 69-75.

(GS) **Green, P.J., Sibson, R.**, Computing Dirichlet Tesselations in the Plane, The Computer Journal, Vol. 21, 1978, pp. 168-173.

(GSe) **Guibas, L., Seidel, R.**, Computing Convolutions by Reciprocal Search, ACM Symp. on Computational Geometry, Yorktown Heights, New-York, June 2-4, 1986.

(GSt) **Guibas, L., Stolfi, J.**, Primitives for the Manipulation of General Subdivisions and the Computation of Voronoi Diagrams, ACM Trans. on Graphics, Vol.4, N° 2, April 1985, pp. 74-123.

(H) **Hermeline, F.**, Triangulation Automatique d'un Polyèdre en Dimension N, RAIRO Anal. Numer., Vol. 16, N° 3, 1982, pp. 211-242.

(K) **Keppel, E.**, Approximating Complex Surfaces by Triangulation of Contour Lines, IBM J. Res. Develop., N° 19, Jan. 1975, pp. 2-11.

(Ki) **Kirkpatrick, D.G.**, Optimal Search in Planar Subdivisions, SIAM journal on Computing, Vol. 12, N°1, Feb. 1983, pp. 28-35.

(Kl) **Klee, V.**, On the complexity of d-dimensional Voronoi diagrams. Arch. Math. 34, 1980, pp. 75-80.

(LS) **Lee, D.T., Schacter, B.J.**, Two Algorithms for Constructing a Delaunay Triangulation, Int. Journal of Computer and Inf. Sciences, Vol. 9, N° 3, June 1980, pp. 219-242.

(MS) **Mairson, H.G., Stolfi, J.**, Reporting and Counting Line Segment Intersections, Dep. Comput. Sci., Stanford Univ., Stanford, CA, 1984.

(NP) **Nievergelt, J., Preparata, F.P.**, Plane-sweeping Algorithms for Intersecting Geometric Figures, Comm. ACM, Vol. 24, N° 10, 1982, pp. 739-747.

(P) **Pauchon, E.**, Numérisation et Modélisation Automatique d'Objets 3-D, Thesis, Université de Paris-Sud, Centre d'Orsay, Sept. 1983 (in French).

(R) **Rogers, C.A.**, Packing and Covering, Cambridge University Press, 1964.

(S) **Shamos, M.I.**, Computational Geometry, Ph.D. Dissertation, Yale University, 1978.

(Sh) **Shantz, M.**, Surface Definition for Branching Contour-Defined Objects, Computer Graphics, Vol. 15, N° 2, July 1981, pp. 242-270.

(SH) **Sloan, K.R., Hrechanyk, L.M.**, Surface Reconstruction from Sparse Data, IEEE conf. on Pattern Recognition and Image Processing, Dallas, Aug. 1981, pp. 45-48.

(SHo) **Sloan, S.W., Houlsby, G.T.**, An Implementation of Watson's Algorithm for Computing 2-Dimensional Delaunay Triangulations, Adv. Eng. Software, 1984, Vol. 6, N° 4, pp. 192-197.

(W) **Watson, P.P.**, Computing the n-Dimensional Delaunay Triangulation with Application to Voronoi Polytopes, Computer Journal, Vol. 24, N° 2, May 1981, pp.167-172.

Imprimé en France
par
l'Institut National de Recherche en Informatique et en Automatique