Sewing Faces : A topological reconstruction of 6-connected objects bounding surfaces in $3D$ digital images

Frédéric H. Mabin *† Catherine Mongenet †

Abstract. We present a new algorithm, called "Sewing Faces". From $3D$ images defined by a block of voxels, this algorithm based on a contour following reconstructs bounding surfaces of 6-connected objects. A bounding surface is a set of faces shared by two voxels : one belonging to the object, the other one belonging to its complement. The topological model used to represent these surfaces consists in a set of such faces which are sewed together in such a way that they define a closed surface, or skin. The reconstructed surface can then be embedded into space to give a surface mesh. We show that the complexity of the algorithm is linear in time and space relatively to the number of faces of the skin.

1 . Introduction

The magnetic resonance imaging (MRI) technique developed in the 80s has deeply contributed to the evolution of medical imaging from $2D$ to $3D$. Using this technique one can get $3D$ digital images of any part of the human body. These images are characterized by $3D$ integer matrices called blocks. Each integer defines a value associated with a volume element or voxel of the image. This value represents the summation of the nuclear magnetic resonance (NMR) signals from the substances found in the voxel. From a digital imaging processing point of view such images can be seen as grey scale images.

Assuming that preliminary image processing has been realized such that the objects or organs described by the $3D$ image are "distinguishable" [3], we want to reconstruct bounding surfaces separating the discretized objects from the background.

Many research works have focused on this problem. The various existing methods can be classified using several criteria. The first one, based on the Van Gelder et Wilhelms classification [10] distinguishes the *methods by approximation* from the *exact methods.* The methods by approximation reconstruct a bounding surface by interpolating the discretized data while the bounding surface reconstructed by the exact methods is composed of faces shared by a voxel of the object and a voxel of the background. A second classification criterion is based on the type of scan used to determine the surface. The surface reconstruction can be realized either by a complete search among all the voxels of the block or by a boundary following for which only the voxels of the object "boundary" are scanned. The boundary following approach yields more efficient algorithms whose time complexity is proportional to the number of voxels of the boundary instead of the number of voxels of the whole block. A third classification criterion is related to the nature of the synthesized information. It can be exclusively geometric, i.e. reduced to the list of the faces defining the bounding surface. On the other side, the information can be both geometric and topological, i.e. composed of a list of faces enriched with topological information stating how these faces are connected together. In this later case the closed bounding surface delimiting the $3D$ object is called a *skin.*

*FORENAP - Centre Hospitalier, Service du Dr J.P. MACHER, F-68250 ROUFFACH, E-mail : Frederic.MABIN@forenap.asso.fr

†Université Louis Pasteur de Strasbourg, Laboratoire ICPS, Pôle API, Boulevard Sébastien Brant, F-67400 ILLKIRCH, E-mail : mongenet@icps.u-strasbg.fr

The first methods have been developed in the context of visualization and are therefore exclusively geometric. However the determination of both geometric and topological information is required for many applications such as the optimized display of a bounding surface, the distortion of a surface, the transformation of a surface into a surface mesh, the derefinement of a surface by merging adjacent coplanar faces, the reversible polyhedrization of discretized volumes.

From the geometric information defining a bounding surface, it is of course possible to recover the topological information, i.e. the neighbourhood information between faces. For each face, one have to scan all the other faces defining the bounding surface in order to find its adjacent faces, i.e. the ones which share one edge with it. If the surface contains n faces then this topological reconstruction is $O(n^2)$. To avoid this quadratic operation the topological information must be collected together with the geometric information.

Among the many works on boundary reconstruction the most well-known is probably the Marching Cube developed by Lorensen and Cline [4]. It is a method by approximation which scans all the voxels of a $3D$ block and builds a triangulation of the bounding surface. There are cases where the Marching Cube does not work well : ambiguous voxels may result in holes on the bounding surface. Various extensions of the method have been proposed, either by defining a heuristic to solve ambiguous cases [10] or by reducing the number of generated triangles [7]. Faster reconstructions have been developed. Some are based on parallelized versions of the algorithm [6]. Others use the octree abstract data type [11] which reduces the number of scanned voxels. The Marching Cube and its different extensions reconstruct only the geometric information related to a bounding surface. They do not synthesize any topological information.

An alternative to the Marching Cube is the method proposed by Artzy et al. [1]. It is an exact method which extracts the faces shared by a voxel of the object and a voxel of the background. It is based on a boundary following but does not extract any topological information.

The method we propose in this paper is an exact method (it extracts faces belonging to both a voxel of the object and a voxel of the background) based on a boundary following. Moreover it is both geometric *and* topological since the faces are extracted together with their adjacence relations. These bounding surfaces called *skin* are closed, they define 6-connected objects. This method is founded on the regular surfaces proposed by Rosenfeld [9] who studied their topology without developing a reconstruction method.

The paper is organized as follows. Section 2 recalls standard notions of $3D$ imaging and topology and introduces new notions related to $3D$ objects skin. We show in Section 3 that the geometric and topological reconstruction of the bounding surface of a 6-connected $3D$ object is always possible using our boundary following. The algorithm is briefly described in Section 4 while Section 5 presents the transformation of the bounding surface into a surface mesh. The complexity analysis of the algorithm is conducted in Section 6. Finally Section 7 discusses some currently under development extensions of this method.

2 . Digital imaging notions

We first recall some basic notions in digital geometry [2] and in $3D$ imaging [8]. We then introduce some new definitions and use topological results on regular surfaces introduced by Rosenfeld [9].

Definition 1 . — *Let G be a subset $[0,X] \times [0,Y] \times [0,Z] \in \boldsymbol{N}^3$; $X, Y, Z > 0$. G is a $3D$ finite* ***grid*** *associated with a coordinates system R. The points of G, called* ***vertices****, are defined in R by their* ***integer coordinates****. From the vertices of a grid, we construct the edges, faces and voxels. Each face is numbered from 0 to 5 relatively to a voxel which contains it. A face numbered with i ($i \in [0,5]$) is called a* ***face***

*i or a **i-face**. The centers of all the voxels built on grid G constitute a **dual grid** $G^d = [0, X-1] \times [0, Y-1] \times [0, Z-1] \in \boldsymbol{N}^3$. We associate with G^d a dual coordinates system R^d in which the voxels are defined.*

Definition 2 . — *Two edges sharing exactly one vertex are **adjacent**. Two faces sharing exactly one edge are **adjacent**. Two voxels sharing exactly one face are **6-adjacent**, or **neighbours**[1]. Two voxels sharing at least one edge are **18-adjacent**[2].*

Definition 3 . — *If voxel v shares one of its six faces f with voxel v', we say that f is the face by which v **sees** v', or that **the neighbour of** v **by face** f is v'. The set of voxels which share one face with v is called the **neighbourhood** and is denoted by $N(v)$. We have $0 \leq$Card(N(v))≤ 6.*

Definition 4 . — *A sequence of voxels $u_0, ..., u_k$ such that $\forall i \in [0, k[$, voxels u_i and u_{i+1} are 6-adjacent (respectively 18-adjacent), is a **6-path** (respectively a **18-path**). Two voxels v et v' between which a 6-path (respectively a 18-path) exists are **6-connected** (respectively **18-connected**).*

Definition 5 . — *A **block** B is a function from G^d to a **domain of values** $D(B)$. We usually have $D(B) = \{0, 1\}$ for **binary blocks**, or $D(B) = [0, V] \subset \boldsymbol{N}$ for **intensity blocks**.*

In the case of binary blocks, the points of G^d can be partitioned into two sets : the **object** (denoted O) and the **background** (denoted $\overline{O}$). In the other cases, if the user defines a subset of $D(B)$ as being the set of values representing the object, we can always come back to the binary case.

Definition 6 . — *A set of voxels for which each pair defines 6-connected voxels is **6-connected**.*

[1] Two edges (respectively faces or voxels) sharing more than one vertex (resp. edge or face) define one and only one edge (resp. face or voxel).

[2] Two voxels which are 6-adjacent are 18-adjacent, the inverse is false.

We only consider blocks whose background and object are 6-connected. It means that there is only one object per block and that the object enclose no hole. If the object or the background contain several 6-connected components, we run the algorithm on each surface to extract. The final number of visited voxels will still remain lower or equal to the total number of voxels in the block.

Let us now introduce new notions used for the Sewing Faces algorithm. In the first place, we want to define the "thick contour" of an object. This contour is composed of a set of 18-connected voxels called the **18-boundary**. To determine the voxels of the 18-boundary, we introduce some specific notions and especially the one of **voxel type**.

Definition 7 . — *The **type** of voxel v is defined as follows :*

- *type$(v) = 0 \iff v \in \overline{O}$. We say that v is **outside** of the object, or is a **0-voxel**;*
- *type$(v) = 1 \iff v \in O$ and Card(N(v))= 6 and $\forall\, v' \in N(v), v' \in O$. We say that v is **inside** the object, or is a **1-voxel**;*
- *type$(v) = 2 \iff v \in O$ and either $\exists\, v' \in N(v)\ /\ v' \in \overline{O}$ or Card(N(v)) < 6. We say that v is in the **18-boundary** of the object, or is a **2-voxel**.*

Definition 8 . — *The **18-boundary** of G^d, is the set denoted by $\delta(G^d)$ of all the 2-voxels of G^d.*

In a second place, we focus on the bounding surface, i.e. on the "contour without thickness" of an object. It is composed of a set of adjacent faces coming from the thick contour, i.e. from voxels of the 18-boundary. Each face of a voxel of the 18-boundary shared by one voxel of the background as well as each face of a voxel of the 18-boundary shared by no other voxel (in the case of a voxel located on the circumference of the $3D$ block) belongs to the bounding surface. Such a face is called a **surfel**.

Definition 9 . — *Let $v = \{f_0, ..., f_5\}$ be a 2-voxel of G^d whose neighbour by face f_i ($i \in [0, 5]$) is either in the background, or* *does not exists (i.e. is out of G^d). Face f_i is called a **surfel**. The set of all the surfels of voxel v is denoted by $\Phi(v)$.*

We want to analyze and extract the topology of the surfels. Since the object is 6-connected, two surfels are adjacent by one edge according to three modes introduced by Rosenfeld [9]. We call these modes ***half-sews***.

Definition 10 . — *Two 2-voxels which are 18-adjacent but not 6-adjacent (i.e. they share only one edge) and which neighbourhoods intersection contains exactly one voxel of the background and one voxel of the object called **support voxel**, are said to be **binded**.*

Figure 1 shows two binded voxels and their support voxel : the three grey voxels belong to the object, voxels v et v' are binded, voxel u is their support voxel and the voxel drawn in dotted lines is in the background. The black faces belong to the skin.

Definition 11 . — *The three types of **half-sew**, or **1/2-sew**, between surfels of the skin that are adjacent by exactly one edge are :*
1/2-1-sew : *between two surfels of one 2-voxel of the 18-boundary;*
1/2-2-sew : *between two surfels belonging to two neighbouring 2-voxels of the 18-boundary;*
1/2-3-sew : *between two surfels belonging to two binded voxels. We will say that these two voxels **induce** some 1/2-3-sew on their support voxel, or on the edges of their support voxel.*

Two *1/2-sews* are needed to make one sew. If surfel f is sewed (by one *half-sew*) to surfel f', then surfel f' must be sewed (by another *half-sew* of the same type) to surfel f. The type of a sew is the type of its two *1/2-sews.* We distinguish between 1-sews, 2-sews and 3-sews as shown in Figure 2.

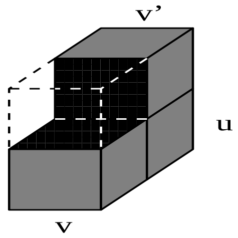

Figure 1: Two binded voxels.

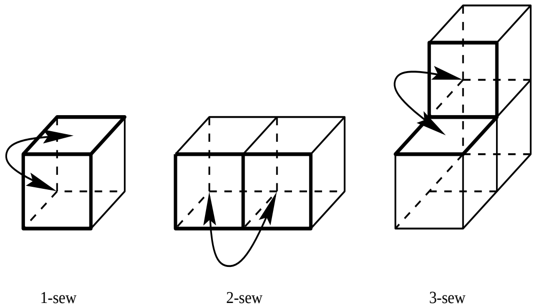

Figure 2: Three types of sews.

Property 1 . — *The 18-boundary is 18-connected, but not necessarily 6-connected.*

Proof. 18-connectivity of the 18-boundary is obvious. To prove that the 18-boundary is not necessarily 6-connected, let us consider a counter example. Figure 3 shows an object composed of seven voxels. The 18-boundary of that object is composed of all voxels of the object, except for the one in the middle : this set of six voxels is not 6-connected. □

Our objective is to extract the surfels of an object by examining as few voxels as possible. Surfels being faces of 2-voxels, it is sufficient to examine the voxels of the 18-boundary to catch all surfels. A way to examine all the voxels of the 18-boundary, is to examine first one voxel and then all of its 6-neighbours which are also 2-voxels. And so on. However we know from Property 1 that some 2-voxels may be forgotten when examining the voxels of the 18-boundary with such a strategy. One way to avoid missing voxels would be to analyze, for a given voxel of the 18-

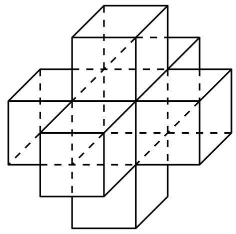
Figure 3: Object composed of seven voxels.

boundary, all its 18-adjacent voxels belonging to the 18-boundary. This strategy would require the scanning of 3 times more voxels. To prevent this inefficiency we use another strategy which relies on the 6-connectivity. It uses the notion of 6-boundary which is the 6-connected "thick contour" of the block including the missing voxels called the binding ones. We will see that apart from the 6-connectivity, the 6-boundary allows us to sew all the surfels of the surface.

Definition 12 . — *Let v be a voxel of type 1 and v_0, v_1 be two binded voxels of $\delta(G^d)$ such that $v \in N(v_0) \cap N(v_1)$. Voxel v is called a **binding voxel**[3].*

Definition 13 . — *The **6-boundary** of G^d, denoted by $\Delta(G^d)$, is the union of the 18-boundary $\delta(G^d)$ and of the set of all the binding voxels.*

Definition 14 . — *Let G be a grid. We call **skin** a triplet $S = (B, F, C)$ where :*
• *B is a block from G^d to $D(B)$;*
• *F is the set of all the surfels of G, such that $\forall f \in F$, $\exists$ **exactly four distinct ordered pairs** $(e_0, f), ..., (e_3, f) \in C$;*
• *C is a set of ordered pairs (e, f), called **half-sews** or **1/2-sews**, where $e \in f$ is an edge of G and $f \in F$ is a face of G, and such that $\forall\ (e, f) \in C$, $\exists\,!\ (e, f') \in C\ f' \neq f$. Set $\{(e, f); (e, f')\}$ is called **sew**, or **sew of edge** e, or **sew between faces** f **and** f'.*

3 . Minimal voxel set to scan

We now state some results guaranteeing that :
• the 6-boundary is 6-connected and can be scanned according to the strategy presented in Section 2;
• each sew can be determined by handling one voxel of the 6-boundary and if necessary its neighbours that also belong to the 6-boundary;
• each voxel of the 6-boundary yields at least one sew.

To prove the 6-connectivity of the 6-boundary, we use the following property.

Property 2 . — *Let G be a grid and X be any 6-connected subset, i.e. with holes or not, of voxels of the object. Let v be a type 1 voxel that is not a binding voxel. $X - \{v\}$ is still 6-connected.*

Proof. To prove that $X - \{v\}$ is 6-connected, we must establish that for any 6-path going through v, there exists a "substitution" 6-path in $X - \{v\}$. Let $u_0, ..., u_i, v, u_{i+1}, ..., u_n$ be any 6-path in X going through v. Let $v' \in N(u_i) \cap N(u_{i+1})$. Since v is 6-adjacent to u_i and u_{i+1}, v' is 18-adjacent to v. Moreover, by definition of a binding voxel, since v is a 1-voxel and is not a binding voxel, the voxels which are 18-adjacent to v are in the object (i.e. of type 1 or 2). Therefore voxel v' is in the object. Path $u_0, ..., u_i, v', u_{i+1}, ..., u_n$ is therefore a 6-path in $X - \{v\}$. ▫

Theorem 1 . — *Let G be a grid. The 6-boundary $\Delta(G^d)$ is 6-connected.*

Proof. The 6-boundary is obtained by deleting from the object all the 1-voxels apart from the ones which are binding voxels. After removing one such voxel, we obtain a voxel set with a hole. Nevertheless, Property 2 guarantees that such a removal will not disconnect the final set. The 6-boundary is therefore 6-connected. ▫

Definition 15 . — *Let G^d be a grid and v be a voxel. $N'(v)$ is the subset of neighbourhood $N(v)$ such that $\forall\ v' \in N(v)$, $v' \in N'(v) \iff v' \in \Delta(G^d)$.*

Theorem 2 . — *Let G be a grid and $S = (B, F, C)$ be a skin. For any sew $\{(e, f); (e, f')\} \in C$, $\exists\ v \in \Delta(G^d)$ and $\exists\ v_0, v_1 \in \{v\} \cup N'(v)$ such that $f \in v_0, f' \in v_1$.*

[3] A binding voxel is a particular case of support voxel.

Proof. Let $cc = \{(e, f); (e, f')\}$ be a sew. Let us distinguish three cases corresponding to the three possible types of sews for cc :

1-sew : faces f and f' belong to a same 2-voxel v_x which is included by construction in the 6-boundary $\Delta(G^d)$. Let $v = v_x, v_0 = v_x$ and $v_1 = v_x$, we have $v_x \in \{v_x\} \cup N'(v_x)$;

2-sew : faces f and f' belong to two neighbours of type 2 v_x and v_y which are included by construction in the 6-boundary $\Delta(G^d)$. Let $v = v_x, v_0 = v_x$ and $v_1 = v_y$. Since $v_y \in N'(v_x)$, we have $v_x, v_y \in \{v_x\} \cup N'(v_x)$;

3-sew : faces f and f' belong to two voxels of type 2 v_x and v_y which share only one edge. Faces f and f' being sewed, voxels v_x and v_y are binded. The intersection of $N'(v_x)$ and $N'(v_y)$ contains exactly one voxel v_z of the object : their support voxel. Two cases arise for v_z :

- v_z is of type 1. In this case v_z is a binding voxel belonging by construction to $\Delta(G^d)$;
- v_z is of type 2. In this case v_z also belongs by construction to $\Delta(G^d)$.

In the two cases, let $v = v_z$, $v_0 = v_x$ and $v_1 = v_z$. Since $\{v_x, v_y\} \subset N'(v_z)$, we have $v_x, v_y \in \{v_z\} \cup N'(v_z)$. □

Theorem 3 . — *Let G be a grid and $S = (B, F, C)$ be a skin. For any voxel v of the 6-boundary $\Delta(G^d)$, there exists one 1/2-sew (e, f) of C such that* $f \in \bigcup_{v' \in \{v\} \cup N'(v)} \Phi(v')$.

Proof. Let v be a voxel of $\Delta(G^d)$. Let us distinguish between two cases corresponding to the type of v :

1-voxel : by definition, there exist two 2-voxels v_1 and v_2 in $\Delta(G^d)$ whom v is the support voxel and such that $\exists\ f_1 \in \Phi(v_1),\ f_2 \in \Phi(v_2)$ and e an edge of G^d such that $e \in f_1 \cap f_2$. Let us consider for instance the *1/2-sew* (e, f_1) of C. We have $f_1 \in \bigcup_{v' \in \{v\} \cup N'(v)} \Phi(v')$;

2-voxel : v contains at least one surfel. Let $f \in \Phi(v)$ and (e, f) one *1/2-sew* of C. We have $f \in \bigcup_{v' \in \{v\} \cup N'(v)} \Phi(v')$. □

4 . Construction of the skin

The skin of an object is the set of all the surfels separating the inside from the outside of the object, and sewed together according to one of the three sew types previously described. Since we suppose that there is no hole in the object, the skin is made of only one bounding surface "object / background".

The skin is well-defined by scanning the voxel set of the 6-boundary and by determining the sews. Theorem 1 guarantees this can be done by analyzing and treating the 6-neighbourhood of each voxel of the 6-boundary. Recall that any face i of a 2-voxel v is added to the skin if and only if the neighbour of v by face i is of type 0. The scanning of all the voxels of the 18-boundary is therefore sufficient to create all the surfels of the skin. But determining all the sews (particularly these of type 3) requires the scanning of the 6-boundary. This is guaranteed by Theorem 2.

The algorithm needs beforehand one 2-voxel v_f. This voxel is either given by user, or determined using a dichotomous search algorithm of the 2-voxels in G^d. This voxel is the starting point of the algorithm : its neighbourhood is inspected, then the neighbourhood of its neighbourhood and so on. If the object or the background include more than one 6-connected component, we must run the algorithm on all the surfaces we want to extract. For each run of the algorithm, a new initial voxel v_f must be given or determined.

4.1 . Principle of the algorithm

The algorithm reconstruct progressively the skin defined by triplet $S = (B, F, C)$. It successively scans all the voxels of the 6-boundary. For each of them it examines those of its six neighbours that also belong to the 6-boundary. For each of them, it detects the surfels they generate and it creates the necessary sews. For efficiency reasons, the algorithm is iterative. It uses a stack of 2-tuples. Each 2-tuple or **call** is composed of two neighbour voxels : a **previous voxel** denoted by v_p and a **current voxel** denoted by v_c. Such a call is in the stack if it has been pushed during a previous iteration step when voxel v_p was the current one. We call **co-calls** the set of all the calls generated by one voxel, i.e. having all the same v_p. Co-calls are altogether pushed, the ones on top of the others. The core of the algorithm which is fully described in [5] is :

SewingFaces $(B\ : G^d \to D(B),\ v_f) =$
  $stack \leftarrow$ **NewStack** ()
  $S = (B, F, C) \leftarrow$ **NewSkin** ()
  Treat $((null, v_f),\ S)$
  Co-Calls $(v_f,\ S,\ stack)$
  While (**NotEmpty** $(stack)$) **Do**
    $(v_p, v_c) \leftarrow$ **Pop**$(stack)$
    Treat $((v_p, v_c), S)$
    Co-Calls $(v_c,\ S,\ stack)$
  Done.

A call may be poped a long time after it has been pushed and a voxel may be the current voxel of more than one call. When a call is poped, it may therefore involve a current voxel which has already been treated. In this case, the treatment of this call is simplified : for instance if the voxel is of type 2 its surfels have already been detected and their 1-sews have already been determined. Such a voxel must not generate its co-calls since they have already been generated. The only authorized calls have the following profiles : *(previous voxel of type 1, current voxel of type 2)*, *(previous voxel of type 2, current voxel of type 1)*, *(previous voxel of type 2, current voxel of type 2)*. A 1-voxel may only be a binding voxel and must not cause any call towards another 1-voxel, in order to avoid the unnecessary exploration of the interior of the object. A call of profile *(previous voxel of type 1, current voxel of type 1)* is therefore forbidden.

From the study of the three sew types (cf. Figure 2), it arises that :

- 1-sews are made by scanning only one 2-voxel;
- 2-sews are made by scanning two 2-voxels which are neighbour;
- 3-sews are made by scanning three voxels : two binded 2-voxels and their support voxel.

Chronologically, each call is at first pushed on the stack, then it is later poped to be treated and finally it produces its own calls. During the treatment of a call, four stages must be realized :

1. detection of the surfels of the current voxel;

2. 1-sews between surfels of the current voxel;

3. 2-sews between surfels of the previous voxel and surfels of the current voxel;

4. *1/2-3-sews* of surfels of the current voxel, i.e. *1/2-3-sews* induced by the current voxel on the previous voxel. In this case the previous voxel is performing the role of the support voxel.

4.2 . Treatment of a call

Let us consider any call (v_p, v_c). Voxel v_c may have already been treated. In this case, stages 1 and 2 previously indicated must not be performed. Notice that voxel v_p has necessarily been treated, otherwise it would not have pushed that call. Stages 3 et 4 can (and must) always be realized. The treatment of a call consists therefore in the following function.

Treat $((v_p, v_c), S = (B, F, C)) =$
  If v_c has never been treated **Then**
    Sew1 $(v_c,\ S)$
  EndIf
  Sew2 $(v_p,\ v_c,\ S)$
  Sew3 $(v_p,\ v_c,\ S)$.

Function Sew1 determines the surfels of v_c. For each surfel s_i it scans all its adjacent faces f_j and checks if s_i and f_j define a 1-sew. In that case the 1-sew is added to the set of sews C.

The 2-sews between surfels of v_p and v_c are determined by function Sew2. This function scans the pairs of faces (f_{p_i}, f_{ci}), and if both faces are surfels then it creates the corresponding 2-sew. Notice that this function is symmetrical, i.e. Sew2(v_p, v_c, S) and Sew2(v_c, v_p, S) realize exactly the same modifications of S.

The *1/2-3-sews* induced by v_c on v_p are realized by function Sew3. One call of this function only detects *1/2-3-sews*. To realize a full 3-sew between two surfels of two binded voxels v_{c1} and v_{c2}, the two function calls (v_p, v_{c1}) and (v_p, v_{c2}), where v_p is the support voxel of v_{c1} and v_{c2}, must be poped and treated. This function is not symmetrical, i.e. Sew3(v_p, v_c, S) and Sew3(v_c, v_p, S) do not realize the same modifications of S. In the first case, the *1/2-3-sews* created are these induced by the surfels of v_c on some edges of v_p. In the second case, the *1/2-3-sews* created are these induced by the surfels of v_p on some edges of v_c. We will see that this not symmetrical aspect must be taken into account in the production of co-calls of v_c.

4.3 . Co-calls production

When the treatment of a call (v_p, v_c) is done, v_c must produce a co-call towards each of its neighbour voxels, apart from : **(1)** the 0-voxels, **(2)** the 1-voxels if v_c is itself of type 1, **(3)** its previous voxel v_p.

Points 1 and 2 allow to produce only authorized profile calls. Point 3 is used to avoid unterminated run of the algorithm. However this point rises a problem : some *1/2-3-sews* may be missing. Hence to fully realize a 3-sew involving v_p, v_c and a third voxel two calls to Sew3 are required : Sew3(v_p, v_c, S) and Sew3(v_c, v_p, S). However, point 3 forbids this later call to ensure the terminaison. To remedy this problem, these *1/2-3-sews* are realized by v_c while producing its co-calls.

5 . Embedding of the skin

We have realized so far only topological operations (adding faces into a set of surfels, adding sews into a set of sews), without any consideration of real coordinates of the vertices. Therefore the extracted surface is a "topological surface". To transform it into a “geometrical surface”, i.e. into a $2D$ mesh of the object bounding surface, we embed the skin. A surfel embedded in the $3D$ space becomes a **facet**. To convert all surfels into facets, a starting point is required : the real coordinates of the four vertices of a given surfel f. From the face type of face f and from its sews types, it is easy to deduce from what j-face arises each of the four surfels sewed with f. We can compute coordinates of the four surfels adjacent to f. And so on. We thus obtain the real coordinates of all the surfels and we get all the facets. If the three dimensions of the basic parallelepiped representing one voxel are integer values, embedding of the skin does not require any computation with real values.

6 . Results and discussion

We have run Sewing Faces on synthetics $3D$ blocks, of growing size $n \times n \times n$, representing digital balls of growing radius n. The tests were realized on a Intel Pentium 133 Mhz, under Linux 2.0.0. with 32 Mo of memory. Experimental results are summarized in Table 1. User times are measured in seconds and memory in Kbytes.

The method has been developped under the following objectives : **(1)** to prove the theoretical sturdiness, **(2)** to reach a near optimal complexity in time and space, **(3)** to ensure practical reliability.

In Section 3, point 1 is proved. The stated theorems ensure that the choice of the scanned voxels, the way to scan them and the way to create all sews allow, in all the

n	vox. scanned	% 1-vox.	# surf.	user time	time/surf.	mem.	mem./surf.
85	29106	34.80	34710	1.28	0.00003702	2890	0.083261
95	36378	34.87	43254	1.61	0.00003733	3510	0.081149
105	44426	34.97	52782	1.97	0.00003741	4318	0.081808
115	53426	35.01	63270	2.35	0.00003714	5174	0.081777
125	63074	35.04	74694	2.77	0.00003708	5896	0.078935
135	73634	35.10	86958	3.32	0.00003823	6844	0.078705
145	84866	35.00	100302	3.75	0.00003743	7986	0.079620
155	97058	35.14	114582	4.30	0.00003752	9086	0.079297
165	110058	35.16	129750	4.90	0.00003776	10328	0.079599
175	123810	35.16	145854	5.55	0.00003805	11596	0.079504
185	138426	35.19	162846	6.18	0.00003794	13030	0.080014
195	153722	35.20	180870	6.92	0.00003825	14438	0.079825
205	169946	35.25	199878	7.63	0.00003817	15968	0.079889

Table 1: Experimental results.

cases, to reconstruct the skin. There is no exception, the only condition is that the object must be 6-connected.

To prove point 2, we first send back the reader to the different algorithms. Functions Sew1, Sew2 and Sew3 are the elementary functions. They are called a linear number of times relatively to the number of voxels of the 6-boundary. Except for the binding voxels (which represent in average 35% of the whole voxels of the 6-boundary), all the voxels of the 6-boundary generate some surfels. The number of calls to the elementary functions is therefore linear with the number of skin surfels. This is corroborated by the column *time/surf* of the experimental results table.

The basic functions implementation requires data structures that are linear in space with the number of voxels of the 6-boundary. If we disregard binding voxels for the same reasons than above, we deduce that the algorithm is linear in space with the number of skin surfels. It is also corroborated by the column *mem/surf*.

The third point led us, at first, to write the algorithm in Caml-Light, a functional language allowing to write runable specifications and to validate a program. Later on, we wrote again the program in the C language in order to mesure better execution times. From the numerous tests realized, either on real data coming from MRI or on synthetic data, we have verified the reliability of our C release to Sewing Faces.

7 . Conclusion

We have presented in this paper an algorithm for the topological reconstruction of the skin or bounding surface of 6-connected $3D$ objects. This algorithm is based on a boundary following, therefore not all the voxels of the $3D$ block are scanned. The reconstructed surface is exact, i.e. composed of voxels faces without any approximation. It has been shown that the time and space complexity of the method are linear according to the number of surfels of the skin. This method is therefore optimal. Its major interest is that it focuses not only on the geometry of the skin but also on its topology, which is synthesized without increasing the complexity of the algorithm. The skin surfels are sewed together in such a way that each surfel knows its four adjacent surfels. The sewing is realized according to three modes as described by Rosenfeld. A topological extension of the algorithm would consist in building the inclusion tree in order to deal with several objects defined in an image. Due to its topological nature this method may be useful in various image processing applications such as surfaces deformation and derefinement, surface meshes manipulation, reversible polyedrisation of discretized volumes.

Various parallelizations of this method are already under development. One is a sub-block approach. The $3D$ image is decomposed into several sub-blocks that are analyzed in parallel. The partial skin related to each sub-block is determined and an aggregation process reconstructs the whole skin from the partial ones. With such a large-grain parallelization approach one can work on large $3D$ images that are too big to fit in memory. A small-grain parallelization is also studied. It is based on modifications of the data structures, in particular the call-stack.

Acknowledgments. This research is supported by FORENAP, the Foundation for Applied Neuroscience Research in Psychiatry. We wish to thanks Dr J.P. Macher who is the scientific director of FORENAP, Laurent Soufflet and Michel Toussaint of the Computer Science Department of FORENAP for their help.

References

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

[2] Jean-Marc Chassery and Annick Montanvert. *Géométrie discrète*. Traité des Nouvelles Technologies série Image. Hermès, May 1991.

[3] Edward R. Dougherty. *Digital image processing methods*, volume 42 of *Optical Engineering*. Marcel Dekker, Inc., 1994.

[4] William E. Lorensen and Harvey E. Cline. Marching cubes : a high resolution 3d surface construction algorithm. *ACM Computer Graphics*, 21(4):163–169, July 1987.

[5] F. H. Mabin and C. Mongenet. Sewing faces : A topological reconstruction of 6-connected objects bounding surfaces in 3d digital images. (extended version). Research report, Informatique et Calcul Parallèle de Strasbourg, December 1996.

[6] Serge Miguet and Jean-Marc Nicod. A load-balanced parallel implementation of the marching-cubes algorithm. Research Report 95-24, Laboratoire de l'Informatique du Parallélisme, Ecole Normale Supérieure de Lyon, October 1995.

[7] C. Montani, R. Scateni, and R. Scopigno. Discretized marching cubes. March 1994.

[8] Laurent Perroton. Object borders, surfaces and contours in 3d digital images. Research Report 93-28, Laboratoire de l'Informatique du Parallélisme, Ecole Normale Supérieure de Lyon, June 1993.

[9] A. Rosenfeld, T. Yung Kong, and Angela Y. Wu. Digital surfaces. *CVGIP : Graphical Models and Image Processing*, 53(4):305–312, July 1991.

[10] Allen Van Gelder and Jane Wilhelms. Topological considerations in isosurface generation. *ACM Transactions on Graphics*, 13(4):337–375, October 1994.

[11] Jane Wilhelms and Allen Van Gelder. Octrees for faster isosurface generation. *ACM Transactions on Graphics*, 11(3):201–227, July 1992.