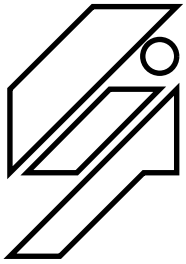


Laboratoire de l'Informatique du Parallélisme

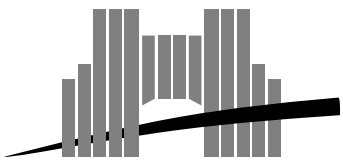
Ecole Normale Supérieure de Lyon
Unité de recherche associée au CNRS n°1398

Volumic Segmentation using Hierarchical Representation and Triangulated Surface

Jacques-Olivier Lachaud
Annick Montanvert

October 95

Research Report N^{O} 95-37



Ecole Normale Supérieure de Lyon
46 Allée d'Italie, 69364 Lyon Cedex 07, France
Téléphone : (+33) 72.72.80.00 Télécopieur : (+33) 72.72.80.80
Adresse électronique : lip@lip.ens−lyon.fr

Volumic Segmentation using Hierarchical Representation and Triangulated Surface

Jacques-Olivier Lachaud
Annick Montanvert

October 95

Abstract

This research report presents a new algorithm for segmenting three-dimensional images. It is based on a dynamic triangulated surface and on a pyramidal representation. The triangulated surface, which follows a physical modelization and which can as well modify its geometry as its topology, segments images into their components by altering its shape according to internal and external constraints. In order to speed up the whole process, an algorithm for pyramid building with any reduction factor allows us to transform the image into a set of images with progressive resolutions. This organization into a hierarchy, combined with a model that can adapt its mesh refinement to the resolution of the workspace, authorizes a fast estimation of the general forms included in the image. After that, the model searches for finer and finer details while relying successively on the different levels of the pyramid.

Keywords: three-dimensional segmentation, deformable model, three-dimensional pyramid, complex topology, triangulated surface, multi-scale

Résumé

Ce rapport de recherche présente un nouvel algorithme de segmentation d'images tridimensionnelles par utilisation de pyramides et de triangulation de surface dynamique. La triangulation, dotée d'une modélisation physique et capable de changer sa topologie, va, en se déformant suivant certaines contraintes, segmenter l'image en ses constituants. Afin d'accélérer le processus, un algorithme de construction de pyramide de facteur de réduction quelconque permet de transformer l'image en un ensemble d'images de résolution progressive. Cette hiérarchisation, couplée à un modèle capable d'adapter la précision de sa maille à la résolution de son espace de travail, permet d'estimer très rapidement les formes générales contenues dans une image. Une fois ceci fait, le modèle recherche les détails de plus en plus petits en s'appuyant successivement sur les différents niveaux de la pyramide.

Mots-clés: segmentation tridimensionnelle, modèle déformable, triangulation de surface, topologie variable, multi-résolution, pyramide tridimensionnelle

Volumic Segmentation using Hierarchical Representation and Triangulated Surface

Jacques-Olivier Lachaud and Annick Montanvert
LIP, ENS-Lyon, URA CNRS 1398
46, allée d'Italie, 69364 LYON Cedex 7
tel: 72.72.85.03 or 72.72.85.86
fax: 72.72.80.80
e-mail: (jolachau, montanv)@lip.ens-lyon.fr

November 9, 1995

1 Introduction

Volumic segmentation has become a major research topic in the last years. This is due to the appearance of three-dimensional data in medical, geological or biological domains. This kind of data can come from either MR, tomography or confocal microscopy. Whereas bi-dimensional segmentation tries to mimic the vision process of the human being, volumic segmentation widens the detection of forms to the reconstruction of complex volumes, which is a difficult operation for our mind.

Unfortunately the analysis of three-dimensional data and the detection of objects inside set a lot of additional problems, like the control of objects with complex topology or the computational cost of the operations in a volumic space. Moreover, at the present time, the means of acquisition introduces noises in the data. Thus direct segmentations are often inadequate and the resort to a deformable model is then essential.

Hence the first purpose has been to evaluate the state of the art in order to develop a deformable model, semi-discrete, dynamical, and possessing a variable topology. Then, in order to overstep the scope of classical segmentation and to accelerate the processing, we associate a scalar continuous field with the image and we introduce the notion of pyramids composed of three-dimensional images. Eventually, we have tested our model and we have measured the savings given by the use of a hierarchical approach. A part of this work is described in **[11]**.

2 Deformable surface

2.1 Overview of the deformable models

A deformable model is a model that follows a general principle: the object is deformed until it minimizes an energy function. The formulation of the minimization can be one of the followings:

- either directly: the energy function is explicit and calculable. We use a mathematical tool, such as least-square method for instance, in order to find the minimum. After this stage, we deduce the parameters of the model.

- or indirectly: the model constrained by internal and external forces evolves until it stops at an equilibrium point. Once this is done, the current parameters minimize the energy function linked with the constraint system.

Thus, for the bi-dimensional case, the *snake* (or active contour) is a deformable curve which segments images using energy minimization **[8]** **[5]**. The *snake* escapes the problems of a global minimization by making a certain number of local minimizations until it reaches a stable point. The least-square method is then applied locally. Because the *snake* can be extended to more general problems, many three-dimensional models are based on this principle.

For the three-dimensional case, the modelizations based on quadrics, superquadrics **[23]** **[1]** and hyperquadrics **[9]** are very popular. These models segment images by computing both local and global parameters. In **[23]**, a set of forces depending on the image and on the model properties is applied in order to direct it toward the desired solution. In **[1]**, the authors compute first some global parameters by a least-square method and then they sharpen the result by adjusting a control-box. Unfortunately these two methods require objects homologous to a sphere for correct results.

Volumes can also be designed by implicit functions (or *blobs*)**[25]**. These active *blobs* behave like bi-dimensional *snakes* because they perform a lot of local minimizations of the model parameters. The main drawback is the computational cost needed to solve non-linear differential equations.

A similar approach is the front propagation **[15]**. The form is defined as an implicit function of $\mathbb{R}^4 \rightarrow \mathbb{R}$. This definition allows the use of any topology in $\mathbb{R}^3$. On the other hand, the computational cost is greatly increased due to the use of an upper dimension.

Deformable meshes are more classical approaches and, in this case, the segmentation is done by constraining the model on its vertices. The mesh can be either a triangulated surface **[17]** **[10]**, a cubic spline **[13]**, or any other structure. Whichever structure is chosen, a set of local constraints is applied to preserve the homogeneity of the model. The problem is then to solve the topological breaks. For the spline case, **[12]** gives a partial solution to this problem.

2.2 A flexible triangulated surface

In order to segment three-dimensional images, we have chosen our model according to two essential criteria: the quickness and the quality of results.

The last approach, the triangulated mesh, seems to be the best choice after the examination of each model. Being a direct extension of the bi-dimensional *snake*, its advantages are quite clear:

- simplicity,
- quickness($\mathcal{O}(N^2)$ vertices for a volumic image with edges of N voxels),
- fast and easy rendering.

To these points we can add the opportunity of extracting features of the object, such as the area and the volume defined by the object, moments and topological informations. Thus the application domain is considerably widen and meets a lot of various domains: segmentation, volume representation, CAD, ...

This model isn't sufficiently efficient for our purpose, so we have improved it with both a physical modelization and a flexible geometry. The first aspect, described in **section 3.2** characterizes the deformation principle of the object under forces built from the image and from internal constraints. The second one, described in **section 3.3** manages a constant coherence between the analytical shape of the object and its geometry. In other terms, the model will always represent a real shape, even if the constraints force it to modify its geometry or its topology.

3 Overview of the model

3.1 Triangulated Mesh

Our segmentation is based on a deformable surface. We define it as a closed oriented triangulated mesh. With this definition the surface always represents the boundary of a real volume. The following rules give a formal definition of a closed oriented triangulated mesh τ:

- τ is a finite reunion of triangles,
- two different triangles have either an empty intersection, an intersection reduced to a vertex or an intersection reduced to an edge,
- any edge belongs to exactly two distinct triangles,
- two locally distinct surface elements can't meet at one vertex,
- the orientation must globally define an inner and an outer part.

Our surface may thus be composed of several connected components, all closed and oriented.

3.2 Physical Aspect

The mesh is assimilated to a dynamic system of particles, which are the vertices of the triangles, following both constraints of other particles and of the environment. Practically, interactions between particles occur only between direct neighbours. **[21]** models a dynamic particle system too, but he doesn't rely on the connections of triangulated meshes in order to find quickly the involved particles.

Hence we use the geometry of our surface to detect the neighbours of the vertices, that is the direct connected ones. We define two internal forces that depend on the neighbouring of each vertex: the force of curvature resistance $\mathcal{F}_c$ which smoothes the shape, and the force of surface elasticity $\mathcal{F}_e$ which spreads localized deformations along the whole surface.

$$\forall S,\ \ \overrightarrow{\mathcal{F}_c(S)} = \lambda_c \cdot \left[\left(\overrightarrow{X(S)X_g(S)} \right) - \sum_{i=0}^{N_S-1} \frac{1}{N_{S_i}} \cdot \left(\overrightarrow{X(S_i)X_g(S_i)} \right) \right] \tag{1}$$

$$\text{where} \begin{cases} \lambda_c \text{ is the coefficient of curvature resistance.} \\ X(S) \text{ are the coordinates of the vertex S.} \\ \overrightarrow{X_g(S)} \text{ is the barycenter of the neighbours of } S. \end{cases}$$

$$\forall S,\ \ \overrightarrow{\mathcal{F}_e(S)} = \sum_{i=0}^{N_S-1} \lambda_e \cdot \left(\left\| \overrightarrow{X(S)X(S_i)} \right\| - d_G \right) \cdot \frac{\overrightarrow{X(S)X(S_i)}}{\left\| \overrightarrow{X(S)X(S_i)} \right\|} \tag{2}$$

$$\text{where} \begin{cases} \lambda_e \text{ is the stiffness coefficient.} \\ d_G \text{ is the global average length of the edges.} \end{cases}$$

These two forces follow the action/reaction principle. The first one brings back vertices to their local tangent plane. It represents the surface tension energy. The second one regularizes the edge lengths along the entire surface. It expresses the binding energy.

Our main purpose is the segmentation of objects, parts of a volumic image. To do so, we express the influence of the image, noted I, as a set of constraints on our model. Physical constraints must be defined everywhere, so we consider our workspace as a continuous scalar field of $\mathbb{R}^3$. Starting from a discrete image I, we compute a potential function from $\mathbb{R}^3$ toward $[0,1]$ noted Π_I. This transformation, described in **section 4.1**, allows us to define two external forces $\mathcal{F}_i$ and $\mathcal{F}_{di}$ derived from the continuous scalar field and intended for different purposes:

$$\forall S,\ \ \overrightarrow{\mathcal{F}_i(S)} = \lambda_i \delta \left(\Pi_0 - \Pi_I(\overrightarrow{X(S)}) \right) \cdot \overrightarrow{X_n(S)} \tag{3}$$

$$\begin{aligned} &\text{Let } \overrightarrow{v_{di}} \text{ be } \Pi_{\nabla_i I}\left(\overrightarrow{X(S)}\right) \cdot \vec{\imath} + \Pi_{\nabla_j I}\left(\overrightarrow{X(S)}\right) \cdot \vec{\jmath} + \Pi_{\nabla_k I}\left(\overrightarrow{X(S)}\right) \cdot \vec{k} \\ &\forall S,\ \ \overrightarrow{\mathcal{F}_{di}(S)} = \delta \left((\lambda_{di} - \mu_{di}) \left(\overrightarrow{v_{di}} \bullet \overrightarrow{X_n(S)} \right) \cdot \overrightarrow{X_n(S)} + \mu_{di} \cdot \overrightarrow{v_{di}} \right) \end{aligned} \tag{4}$$

where $\begin{cases} \lambda_i \text{ is the interaction coefficient and } \Pi_0 \text{ the searched value in the potential function,} \\ \lambda_{di}(\text{resp. } \mu_{di}) \text{ is the coefficient of gradient interaction along } \overrightarrow{X_n(S)}(\text{resp.}(\overrightarrow{X_n(S)})^-), \\ \Pi_F(\vec{X}) \text{ is the interpolation of the discrete image } F \text{ at point } \vec{X}, \\ (\nabla_i I, \nabla_j I, \nabla_k I)(i,j,k) \text{ is the gradient vector at point } (i,j,k), \\ \overrightarrow{X_n(S)} \text{ is the surface normal at vertex } S. \end{cases}$

- The force $\mathcal{F}_i$ is meant to search for the isopotential surface of value Π_0. Its principle is to inflate or deflate locally the model as long as it doesn't fit on the desired isopotential surface. A positive value is expected for the coefficient λ_i if the potential function tends toward one *ad infinitum*, a negative one when it tends toward zero.

- The force $\mathcal{F}_{di}$ is a classical gradient descent. The coefficient λ_{di} modulates it along the surface local normal, μ_{di} along the local tangent plane. There's absolutely no contour tracking or reconstruction, as **[18]** does, but only a direct use of the gradient vector.

Both forces are normalized by the parameter δ (see **section 3.3.1**). δ is the invariant controlling the refinement of the triangulated mesh, i. e. the edge length.

The algorithm carrying out the displacement of the surface can be summarized to an iteration of the following steps:

1. Computing of the internal forces and of the forces deduced from the image for all vertices.
2. Re-sampling of the time scale to limit the vertex displacements.
3. Application of the Dynamic Fundamental Law for each vertex.
4. Effective displacement of the vertices.

We must note that this process expresses only the analytical displacement of the surface (changes of coordinates) and not the intrinsic geometrical or topological modifications.

3.3 Geometrical aspect

Triangulated meshes tend to intertwine and to intersect when they are left on their own. A direct checking over all triangles would be very costly in term of computational time. So **[10]** introduces a global invariant δ which bounds the minimal and maximal sizes of each edge (see **section 3.3.1**). By this way, the local geometrical modifications are made easier and tests over vertex distance are sufficient to detect collisions between pieces of surface. We control topological breaks with the help of the Euler-Poincaré's characteristic (see **section 3.3.2**).

3.3.1 Introduction of an invariant δ

The invariant δ is a positive scalar which determines globally the refinement of the mesh. Formally speaking, it defines three geometrical constraints:

$$\forall (U,V) \text{ couple of neighbouring vertices}, \quad \delta \leq \left\| \overrightarrow{UV} \right\| \tag{5}$$

$$\forall (U,V) \text{ couple of neighbouring vertices}, \quad \left\| \overrightarrow{UV} \right\| \leq 2.5\ \delta \tag{6}$$

$$\forall (U,V) \text{ couple of non-neighbouring vertices}, \quad \frac{2.5}{\sqrt{3}}\ \delta \leq \left\| \overrightarrow{UV} \right\| \tag{7}$$

(5) and (6) express the upper and lower bounds of one edge length. They force the triangulated surface to remain regularly sampled. The violation of (7) expresses a collision between two distinct parts of the surface. We use it to detect and solve topological breaks. The numerical constants introduced in these three equations guarantee the independence of the constraints defined above and assure a correct collision detection. **Section 3.4** describes the geometrical and topological modifications essential to the preservation of these rules.

3.3.2 Use of the Euler-Poincaré's characteristic

Surface theory offers a simple way to classify surfaces according to some topological invariants. **[6]** brings out three distinct fundamental topological invariants for a given surface $\mathcal{S}$: the number of boundary curves denoted $\beta(\mathcal{S})$, the Euler-Poincaré characteristic $\chi(\mathcal{S})$, and the orientability number denoted $q(\mathcal{S})$. These invariants are totally independent of any proper paneling of $\mathcal{S}$. In other words the refinement degree of a surface has no influence on its invariants.

Our kind of triangulated surface allows us to simplify the surface characterization: our surface is closed, hence $\beta(\mathcal{S})$ is nil, and it is an oriented one, so $q(\mathcal{S})$ is nil too. Therefore we can classify all our triangulated meshes according to this only criterion $\chi(\mathcal{S})$. Moreover $\chi(\mathcal{S})$ is easily worked out for that kind of surface. (8) defines it as:

$$\chi(\mathcal{S}) = s(\mathcal{S}) - a(\mathcal{S}) + f(\mathcal{S}), \quad \text{where} \begin{cases} s(\mathcal{S}) \text{ is the number of vertex of } \mathcal{S}, \\ a(\mathcal{S}) \text{ is the number of edge of } \mathcal{S}, \\ f(\mathcal{S}) \text{ is the number of facet of } \mathcal{S}, \end{cases} \tag{8}$$

$$\text{We can add another relation for closed triangulated meshes:} \quad 2\,a(\mathcal{S}) = 3\,f(\mathcal{S}) \tag{9}$$

$$\text{From (8) and (9) we can deduce:} \quad \chi(\mathcal{S}) = s(\mathcal{S}) - \frac{1}{3}\,a(\mathcal{S}) \tag{10}$$

Figure 1 summarizes the four main topology changes for a closed oriented surface. The two others will be discussed later. We recall the variations of $\chi(\mathcal{S})$ generated by these changes. We authorize the surface $\mathcal{S}$ to represent several connected components. In that case its characteristic $\chi(\mathcal{S})$ is the sum of the characteristics of all components of $\mathcal{S}$. Only the global variation of the whole surface $\mathcal{S}$ is examined. We build our transformations with the help of the surface operations defined in [**6**].

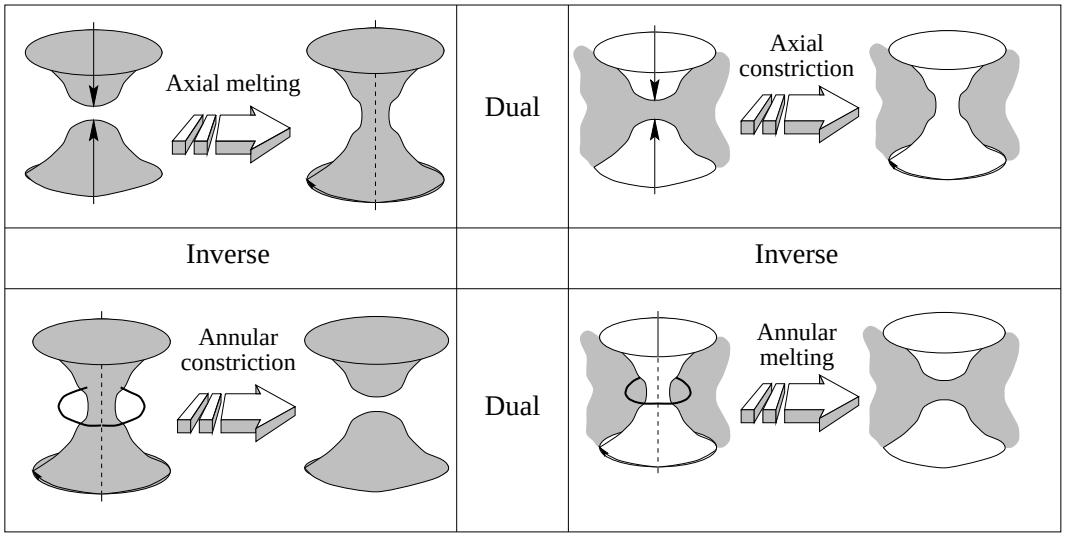


Figure 1: Description of the four main topology accidents for a closed oriented surface of $\mathbb{R}^3$

- **Axial melting**: it occurs when two surface elements, which are not locally neighbours, collide. According to the rules of surface building described in [**6**], the transformation can be split up into the following operations:

  1. a piece of surface is cut out of each surface element. It defines two contours C_1 and C_2,
  2. a bridge is created between the two contours and joins them into one contour C,
  3. the gap of contour C is filled in by a lid, thus closing the surface.

  This process either creates a topological hole in the surface $\mathcal{S}$ or join two distinct connected components into one. The achieved surface $\mathcal{S}_1$ possesses a characteristic $\chi(\mathcal{S}_1)$ as:

  $$\chi(\mathcal{S}_1) = \chi(\mathcal{S}) - 2 \tag{11}$$

- **Annular constriction**: it appears when a piece of surface is too narrow and turns in on itself. The surface splits up into two parts precisely at this location. The following operations modelize this transformation:

  1. a piece of surface is cut out near this location, creating a contour C,
  2. the bridge joining the pieces of surface and builded along contour C is removed from the surface, thus creating two contours C_1 and C_2,
  3. the two gaps defined by these contours are filled in by a lid; the surface is now closed.

This process either eliminates a topological hole or creates a new connected component. The achieved surface $\mathcal{S}_1$ possesses a characteristic $\chi(\mathcal{S}_1)$ as:

$$\chi(\mathcal{S}_1) = \chi(\mathcal{S}) + 2 \tag{12}$$

- **Axial constriction**: it is the dual transformation of the axial melting (by reversing the inside and the outside). The χ variation is so identical to the one of (11).
- **Annular melting**: it is the dual transformation of the annular constriction (by reversing the inside and the outside). The χ variation is so identical to the one of (12).

There are two other topology changes which are independent of the previous ones: the appearance and the disappearance of a surface component. The first one creates a connected component and is directed by the user. It follows (12). The second one eliminates a connected component and occurs when a surface element becomes too small. It follows (11). From all that has been written so far, it is quite easy to demonstrate by recurrence:

$$\forall \mathcal{S},\ \chi(\mathcal{S}) = 2\,(N(\mathcal{S}) - H(\mathcal{S}))\ \text{ where } \begin{cases} N(\mathcal{S}) \text{ is the number of connected components of } \mathcal{S}, \\ H(\mathcal{S}) \text{ is the number of topological holes of } \mathcal{S}. \end{cases} \tag{13}$$

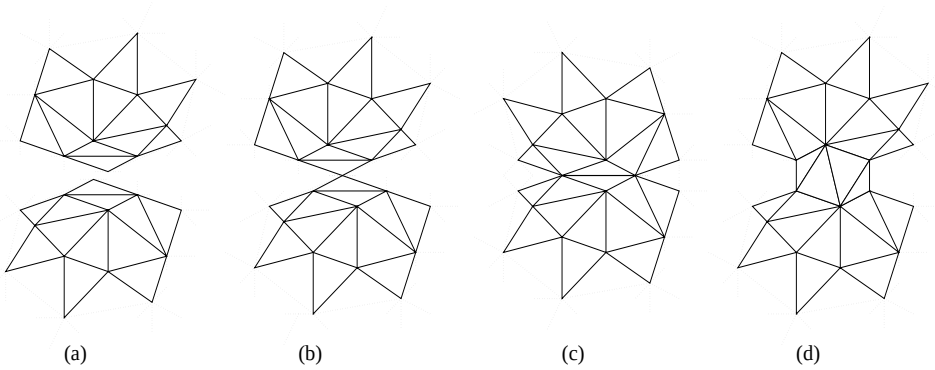


Figure 2: Intermediate forms between two *legal* surfaces: (a) a surface before an axial melting ($\chi = \chi_0$), (b) the same with just one common vertex ($\chi = \chi_0 - 1$), (c) the same with one common edge and two common vertices ($\chi = \chi_0 - 1$), (d) the surface after the axial melting ($\chi = \chi_0 - 2$).

We notice that each transformation modifies the value of χ by two. Strangely the intermediate degenerate surfaces possess an intermediate χ too (see **figure 2**). This property could be relevant and usable to justify the changes of homotopy class.

3.4 Implementation

The triangulated mesh is meant to deform under the action of several forces. The geometrical constraints defined by the invariant (see **section 3.3.1**) may be violated. The geometry of the surface must then be modified in order to eliminate problematic vertices, edges or facets. The verification process of these constraints is performed after each vertex displacement. Arising problems are solved immediately. In **section 3.4.1** we explain how we manage the local coherence on the surface. In **section 3.4.2** we explain how we classify and we solve the collidings between two surface elements.

3.4.1 Solution of local accidents

We use equations (5) and (6) to detect self-intersections. The constraints over each edge length keep the surface from crossing itself.

For every couple of vertices (U, V) with direct common neighbours (S_U, S_V), we denote:

$$d = \left\| \overrightarrow{UV} \right\| \quad \text{and } d' = \left\| \overrightarrow{S_U S_V} \right\|$$

n_U = number of neighbours of U

n_V = number of neighbours of V

The **table 1** describes the verifications of edge lengths for each couple of neighbouring vertices and the correspondingly transformation applied. **Figure 3** describes the geometrical transformations that may be applied on the surface of the mesh.

In the case of a melting, a particular checking is done before trying to melt the two vertices. The following algorithm describes this verification:

MELTING

$\Downarrow$

$n_U = 3$ and $n_V = 3$ $\overset{\text{true}}{\Longrightarrow}$ Tetrahedron to be deleted. End

$\Downarrow$

$n_{S_U} = 3$ $\overset{\text{true}}{\Longrightarrow}$ Vertices S_U is useless, deleted

$\Downarrow$ $\leftarrow\!\!-$ $\hookleftarrow$

$n_{S_V} = 3$ and $n_{S_U} > 3$ $\overset{\text{true}}{\Longrightarrow}$ Vertices S_V is useless, deleted

$\Downarrow$ $\leftarrow\!\!-$ $\hookleftarrow$

Effective melting ...

This first pass eliminates some problematic configurations. Now we are able to check if the melting is just a geometrical problem or a topological one (see **section 3.4.2**).

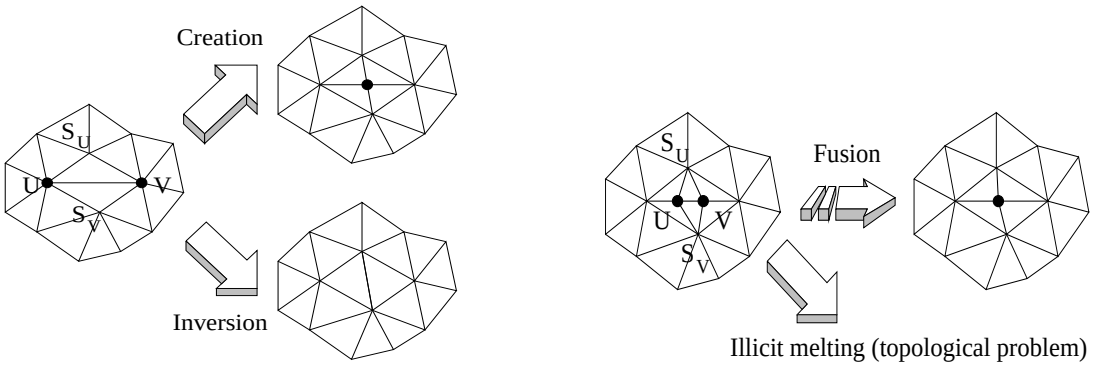


Figure 3: (a) Creation or inversion if U and V are too far away - (b) Fusion or annular problem if U and V are too close.

We now verify that our geometrical transformations don't modify the Euler-Poincaré's characteristic. We use the notations defined in **section 3.3.2** (where $\chi(\mathcal{S}) = s(\mathcal{S}) - a(\mathcal{S}) + f(\mathcal{S})$):

	$\delta \leq d' < 2.0\ \delta$ $n_U > 3$ and $n_V > 3$	$\delta \leq d' < 2.0\ \delta$ $n_U = 3$ or $n_V = 3$	$2.0\ \delta < d'$ $n_U > 3$ and $n_V > 3$	$2.0\ \delta < d'$ $n_U = 3$ or $n_V = 3$
$d < \delta$	MELTING	MELTING	MELTING	MELTING
$\delta \leq d \leq 2.5\ \delta$	INVERSION(*)	nothing	INVERSION(*)	nothing
$2.5\ \delta < d$	INVERSION	CREATION	CREATION	CREATION

(*) INVERSION occurs only if $d > 1.6d'$.

Table 1: Logical array used for deciding the geometrical transformation to apply in order that the surface keeps following the constraints defined by the invariant.

$$\text{If } \mathcal{S} \overset{\text{creation}}{\longrightarrow} \mathcal{S}', \text{ then } \begin{pmatrix} s(\mathcal{S}') \leftarrow s(\mathcal{S}) + 1 \\ a(\mathcal{S}') \leftarrow a(\mathcal{S}) + 3 \\ f(\mathcal{S}') \leftarrow f(\mathcal{S}) + 2 \end{pmatrix} \text{ and } \chi(\mathcal{S}') = \chi(\mathcal{S})$$

$$\text{If } \mathcal{S} \overset{\text{fusion}}{\longrightarrow} \mathcal{S}', \text{ then } \begin{pmatrix} s(\mathcal{S}') \leftarrow s(\mathcal{S}) - 1 \\ a(\mathcal{S}') \leftarrow a(\mathcal{S}) - 3 \\ f(\mathcal{S}') \leftarrow f(\mathcal{S}) - 2 \end{pmatrix} \text{ and } \chi(\mathcal{S}') = \chi(\mathcal{S})$$

$$\text{If } \mathcal{S} \overset{\text{inversion}}{\longrightarrow} \mathcal{S}', \text{ then } \begin{pmatrix} s(\mathcal{S}') \leftarrow s(\mathcal{S}) \\ a(\mathcal{S}') \leftarrow a(\mathcal{S}) \\ f(\mathcal{S}') \leftarrow f(\mathcal{S}) \end{pmatrix} \text{ and } \chi(\mathcal{S}') = \chi(\mathcal{S})$$

The solutions given to local accidents are thus coherent with the predictions imposed by the Euler-Poincaré's characteristic.

3.4.2 Solution of global accidents

We first reduce the problem from four configurations to two: the axial melting and the annular constriction for instance. We can use indeed the duality between the inside and the outside of a volume. Because we represent our volume with a border surface (even if it is an oriented border), there is no distinction to make between an axial melting and an axial constriction or between an annular constriction and an annular melting.

Axial meltings occur when two plane parallel surfaces which are inversely oriented are too close from each other. On a discrete level, two vertices U and V transgress (7). U and V cannot be neighbours, but some of the neighbouring vertices of U may be in the neighbourhood of V. That's why we make distinctions between some of the axial meltings and when (7) is violated, we denote K the number of groups of consecutive common vertices to both U and V (see **figure 4**). We call K the *order of the axial topology rupture.*

What really means the order K? We can interpret it as the kind of topological problem:

0-order rupture it is two non-connected surfaces that are facing each other. It is a simple axial melting or constriction.

1-order rupture it is generally a crossing of vertices on a significantly bent surface. This kind of rupture comes down to a preservation of the local convexity.

2-order rupture and more ... the surface tightens around the two vertices. One (if $K = 2$) or more (if $K > 2$) tunnel-like surface became too narrow on these points. Such a problem is more an annular melting or an annular constriction.

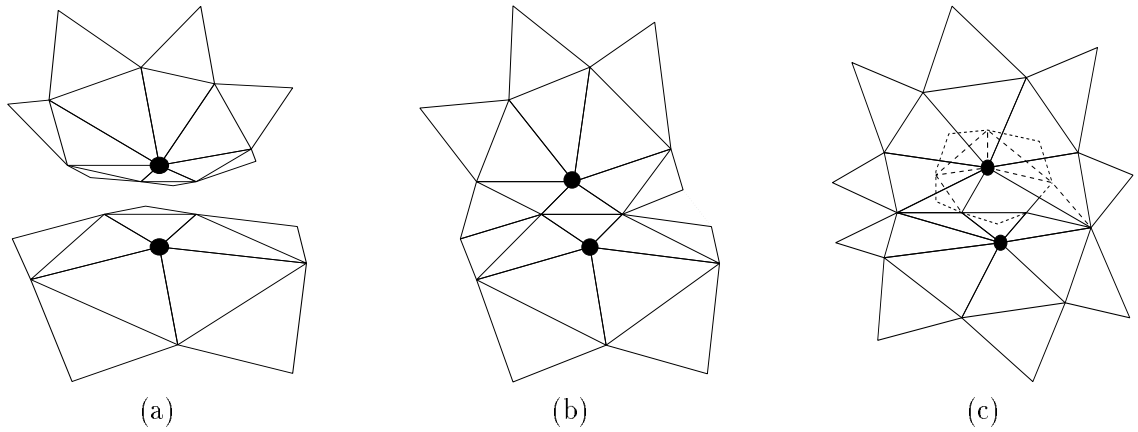


Figure 4: Examples of classification according to the order of the rupture: (a) 0-order rupture - (b) 1-order rupture - (c) 2-order rupture.

A specific solution of all different cases is impossible, but we cannot ignore such cases, even if they are relatively uncommon. So our approach is to transform each rupture into a 0-order rupture before solving the problem.

To do this, we create intermediary vertices between U and its neighbouring vertices and between V and its neighbouring vertices. Even if the vertices and edges created by this process don't follow in general the rules deduced from the invariant, such a transformation reduces all axial ruptures to a 0-order axial rupture (see **Figure 5a**). After this step, we realize a triangulation between these intermediary vertices in order to build a surface (which is internal or external depending upon the surface orientation around U and V) between the two non-connected surfaces (see **Figure 5b**). After having created this tunnel, we remove the vertices U and V just as their links to their ancient neighbourhood. An overall verification is then done on the vertices implied in the topological rupture and some geometrical transformations are realized according to equations (5) and (6).

Remark: We emphasize that this general solution of axial breaks puts aside the problem of a possible annular break for instance (case $K \geq 2$). However it reduces entirely the problem of their detection to one single configuration, which remains its purpose.

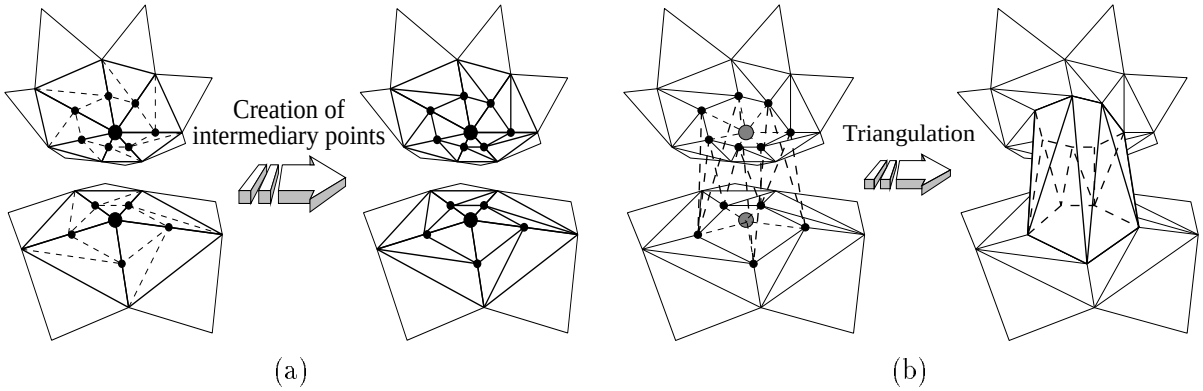


Figure 5: Solution of axial breaks: (a) creation of intermediary vertices - (b) triangulation between the two surfaces and deletion of the two old vertices.

Now annular ruptures can be detected by a simple verification: the presence of what we call *illicit meltings*. They occur when two neighbouring vertices don't represent a surface anymore but

a volume: generally the neighbourhood of the two vertices is just a piece of surface (topologically equal to a square for instance), but in the case of an *illicit melting* the neighbourhood of the two vertices connects to itself on one side (or more) and forms a surface that cannot be shrunk (it defines indeed an essential aspect of the whole surface) (see **figure 6**). A melting of these two vertices becomes impossible without creating an error in the topology of the model.

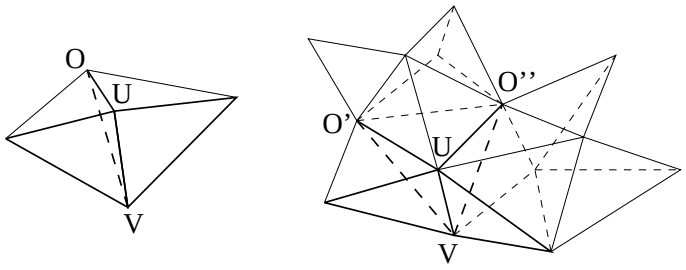


Figure 6: Two examples of illicit meltings.

On a practical level, two vertices define a surface that cannot be shrunk when they possess more than two common neighbours. Two of these common neighbours are compulsory, because they are just the remaining vertices of the two facets containing U and V at the same time. If U and V possess other common vertices, their melting is far more complex than an usual one, because it is a topology break.

The solution of such breaks is based on the dividing of a virtual facet, which is defined by the vertices U, V and the other common vertex. There is no facet at this place, but we divide the surface exactly at this location and then we close the two splitted-in-two surfaces by two facets OUV whose orientations are opposite (see **figure 7**). We reorganize the neighbourhood around the new facets $O_1U_1V_1$ and $O_2U_2V_2$ and we remove the old vertices O, U and V meantime. We can now melt the two edges U_1V_1 and U_2V_2 separately (see also **figure 7**).

One may note that the process has just to be iterated on the critical edge if the illicit melting possesses more than three common vertices at this edge. Speaking on an algorithmic level, as soon as we have to make a fusion, we check if it can be done normally, and, if this is not the case, the annular problem is solved as described previously, and we call back this routine on each divided edge.

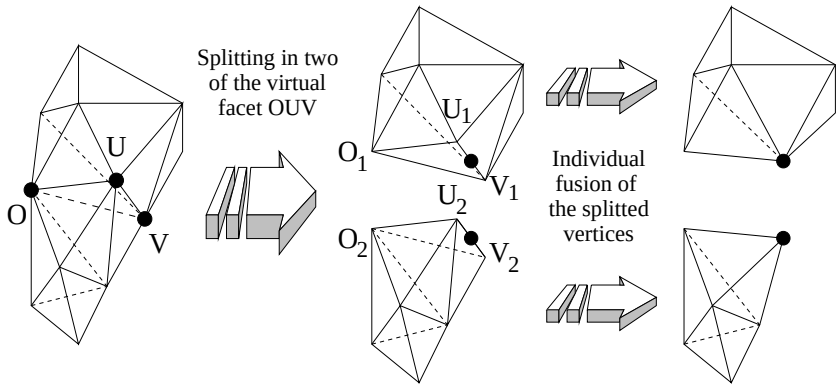


Figure 7: Solution of annular breaks: splitting in two on critical facet then separate fusion.

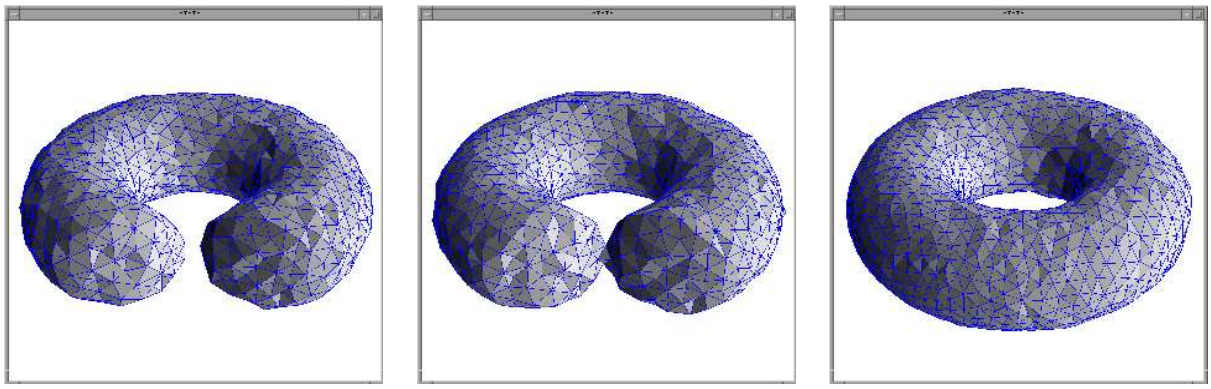
We now verify that our topological transformations modify the Euler-Poincaré's characteristic

according to the predictions given in **section 3.3.2**. We use the notations defined in the same section (where $\chi(\mathcal{S}) = s(\mathcal{S}) - a(\mathcal{S}) + f(\mathcal{S})$) and we denote n_X the number of neighbours of a vertex X:

$$\text{If } \mathcal{S} \overset{\text{interm. vertices}}{\longrightarrow} \mathcal{S}' \overset{\text{triangulation}}{\longrightarrow} \mathcal{S}'', \text{ then}$$
$$\begin{pmatrix} s(\mathcal{S}') \leftarrow s(\mathcal{S}) + n_U + n_V \\ a(\mathcal{S}') \leftarrow a(\mathcal{S}) + 3 * (n_U + n_V) \\ f(\mathcal{S}') \leftarrow f(\mathcal{S}) + 2 * (n_U + n_V) \end{pmatrix} \text{ and } \begin{pmatrix} s(\mathcal{S}'') \leftarrow s(\mathcal{S}') - 2 \\ a(\mathcal{S}'') \leftarrow a(\mathcal{S}') - (n_U + n_V) + (n_U + n_V) \\ f(\mathcal{S}'') \leftarrow f(\mathcal{S}') - (n_U + n_V) + (n_U + n_V) \end{pmatrix},$$
$$\text{and we have } \chi(\mathcal{S}'') = \chi(\mathcal{S}) - 2$$

$$\text{If } \mathcal{S} \overset{\text{illicit melting}}{\longrightarrow} \mathcal{S}', \text{ then } \begin{pmatrix} s(\mathcal{S}') \leftarrow s(\mathcal{S}) + 3 \\ a(\mathcal{S}') \leftarrow a(\mathcal{S}) + 3 \\ f(\mathcal{S}') \leftarrow f(\mathcal{S}) + 2 \end{pmatrix} \text{ and } \chi(\mathcal{S}') = \chi(\mathcal{S}) + 2$$

The implementation of topology ruptures and of geometrical modifications respect fully the predictions given in **section 3.3.2**. The implementation of the axial melting or the axial constriction follows well (11). In the same way the implementation of the annular melting or the annular constriction follows well (12). Moreover, the geometrical modifications applied when using the preservation rules (5) and (6) don't modify the Euler-Poincaré's characteristic of the surface. Therefore all geometrical or topological transformation rules are coherent and transform a closed oriented surface into a surface of the same kind. Thus, the geometrical description of a surface is always coherent to its analytical description. **Figure 8** represents a simple application of dynamical topology: it is the transformation from an object homologous to a sphere to a torus-like object.



(a) (b) (c)

Figure 8: Example of axial break: (a) Object homologous to a sphere - (b) axial melting - (c) torus-like object.

3.5 Initialization of the triangulated mesh

The triangulated surface is initialized with one icosahedron embracing the volumic image in real coordinates or with several icosahedra scattered in the image. The surface is then globally divided (see **section 5.1**) until it follows our geometrical constraints. After that, the surface is free to evolve according to its dynamic and geometrical rules.

4 Image workspace and pyramids

4.1 Transformation toward a continuous scalar field

In **section 3.2** we have defined two external forces $\mathcal{F}_i$ and $\mathcal{F}_{di}$, assuming that the volumic image is a continuous scalar field. However volumic means of acquisition provides discrete image. We propose a transformation from discrete to continuous space by first degree interpolation.

Let $I(i,j,k)$ be our discrete image, with $i = 0 \ldots M-1$, $j = 0 \ldots N-1$, $k = 0 \ldots P-1$. Let μ, ν, π be its real size: it corresponds to the volume really occupied by the image in space.

We determine the continuous potential function $\Pi_I(x,y,z)$, with $x \in [0,\mu[$, $y \in [0,\nu[$, $z \in [0,\pi[$, by interpolating $I()$ with help from:

$$x' = x \cdot \frac{M}{\mu}, \quad y' = y \cdot \frac{N}{\nu}, \quad z' = z \cdot \frac{P}{\pi}, \quad (x',y',z') \text{ coordinates of } (x,y,z) \text{ in } [0,M[\times[0,N[\times[0,P[$$

$$\text{Let } i = \lfloor x' \rfloor, j = \lfloor y' \rfloor, k = \lfloor z' \rfloor, \text{ then, by letting } \alpha = x' - i, \beta = y' - j, \gamma = z' - k$$

$$\left(\begin{array}{ll} \Pi_I(x,y,z) = & (1-\alpha)(1-\beta)(1-\gamma)I(i,j,k) + \alpha(1-\beta)(1-\gamma)I(i+1,j,k) \\ & +(1-\alpha)\beta(1-\gamma)I(i,j+1,k) + \alpha\beta(1-\gamma)I(i+1,j+1,k) \\ & +(1-\alpha)(1-\beta)\gamma I(i,j,k+1) + \alpha(1-\beta)\gamma I(i+1,j,k+1) \\ & +(1-\alpha)\beta\gamma I(i,j+1,k+1) + \alpha\beta\gamma I(i+1,j+1,k+1) \end{array} \right) \tag{14}$$

Figure 9 shows an example of first degree interpolation in a two-dimensional space. Within this context, the potential function is linear when one variable is fixed, and quadratic in the general case. Within the context of three-dimensional space, the potential function is linear with two fixed variables, quadratic when just one variable is fixed, cubic in the other cases.

In this way we obtain a continuous scalar field, but which is not derivable everywhere. That is why we cannot derive directly the gradient from the interpolation. An interpolation of the discrete gradient computed from the discrete volumic image is then preferred.

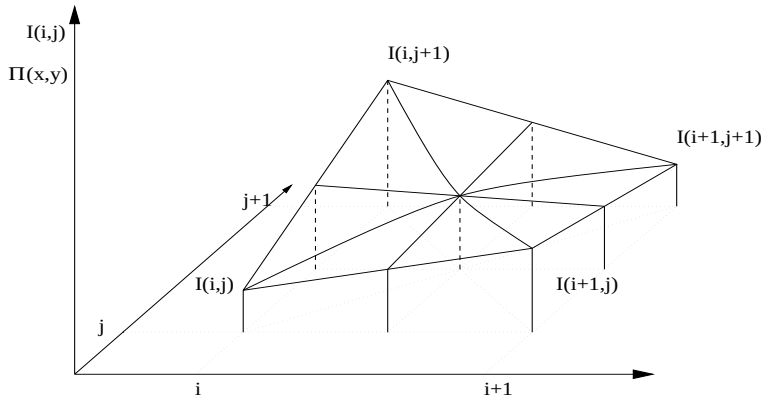


Figure 9: Example of bi-linear interpolation

Unfortunately this transformation toward a continuous field produces some losses of knowledge on the contours and the gradient vectors at these points are distinctly softened. One solution could be the resort to a higher degree interpolation (two or three for instance) despite the considerable increase of computational cost. **Figure 10** illustrates these problems.

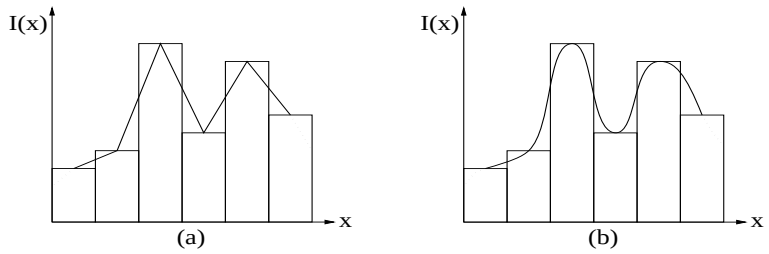


Figure 10: Possible gradient loss with interpolation (one dimensional case): (a) linear interpolation, (b) third degree interpolation.

On the other hand, whatever interpolation degree is chosen, the computed scalar field can be interpreted as a stack of isopotential surfaces. This interpretation justifies the definition of the external constraint $\mathcal{F}_i$ (see **section 3.2**) and thus summarizes the search for forms and contours to a simple search for corresponding isopotential surfaces.

4.2 Multi-scale approach with 3-D pyramids

4.2.1 Motivation

Direct approach of image segmentation is not totally satisfactory. The influence of the potential function derived from image is indeed localized around vertices (according to the definitions of the external forces $\mathcal{F}_i$ and $\mathcal{F}_{di}$ in **section 3.2**) and doesn't make any sense if the triangulated mesh has not a preciseness greater or equal than the discretization preciseness of the three-dimensional image (i. e. its resolution). Two kinds of solutions may be used to solve this problem:

- The first one consists in using a triangulated mesh with a refinement comparable to the one of the image. The surface is then coherent with the frequency domain which it evolves in. The drawback is the need of using a so fine surface that the computational cost is very high.
- The second one consists in using a less finely triangulated mesh, and in taking an interest in a wider area around each vertex. A direct application of this principle is not very useful because the savings done by the use of fewer vertices are offset by the losses due to the longer image examination around vertices.

In order to take advantage of both solutions, we propose to compute once and for all the influence of the image areas at different scales. This mixed solution can be done by the computation of a three-dimensional image pyramid, where all reductions correspond to distinct refinement of the triangulated mesh. The model will exploit the results obtained at inferior resolution levels in order to start the calculation at a finer precision with more efficiency.

4.2.2 Pyramids in image processing

The notion of information contained in an image is closely linked to the resolution the image must be considered. Therefore multi-scale analysis has become common, mainly because it structures image contents by organizing it into a hierarchy; it can be used as well for dealing with objects (entities) as with images.

Pyramidal image representations as proposed by **[22]** have been the first to define and exploit image reduction. However several purposes may be aimed for, among which are the fast computation of parameters, compression, signal decomposition, segmentation, etc **[7]**.

Pyramids of frequency decomposition presented in **[2][3]** provide a set of images at decreasing resolutions which are closed to the visual perception of an observer at increasing distance. The application of a Gaussian kernel filters image high frequencies. After this filtering, a sampling of lower resolution provides the image of higher level. Practically one operator combines the operations of filtering and re-sampling. This process builds the Gaussian pyramid, taking advantage of the fact that a Gaussian kernel does not create any wrong contours. When its size is 5×5 in a two-dimensional space, the waveband is reduced from one octave, hence the sampling frequency is reduced from the same factor.

To get the best out of pyramidal representations we need to define them for volumic image composed of non-cubic voxels and to link them together with our model of surface representation. These points are tackled in **section 5.2**.

4.2.3 Induced problems

Our segmentation is thus based on a dynamical triangulated surface working in a multi-scale space. Its implementation arises some new problems:

- choice of the pyramid: usual pyramids are not always well suited for volumic segmentation. So we have developed an algorithm for creating volumic pyramids of any reduction factor (see **section 4.3**).

- constant appropriateness preservation between the resolution of the current image and the refinement of the triangulated surface: pyramidal approach entails scale variations of the geometrical structure representation. **Section 5.1** shows the way to manage them.

4.3 3-D image pyramids of any reduction factor

In order to segment three-dimensional images, our surface model must follow the discontinuity zones at nearest while embracing homogeneous regions.

A multi-scale treatment of the 3-D image will provide us a coarse-to-fine access to the image. Coupled together with the evolution of our surface model, it will speed up the segmentation process (a critical aspect in 3-D) and assure the process convergence toward a "visually correct" solution.

We first recall the building of a classical Gaussian pyramid.

The successive levels of that kind of pyramid are computed by the convolution of a Gaussian kernel of side 5 pixels (or voxels). It guarantees a low cost filtering without phase translation linked to a reduction factor of two for each image dimension **[4]**.

Let G_0 be the initial image of 3-D voxels and the base of the pyramid. The computation of G_{h+1} (image of level $h+1$ in the pyramid) according to G_h (image of level h in the pyramid) is given by the discrete convolution formula:

$$G_{h+1}(i',j',k') = \sum_{m=-2}^{2} \sum_{n=-2}^{2} \sum_{p=-2}^{2} \omega(m,n,p) \cdot G_h(2i'+m, 2j'+n, 2k'+p) \qquad (15)$$

where ω is a Gaussian convolution kernel of size 5 voxels: $\left(\frac{1}{16}[1\ 4\ 6\ 4\ 1]\right)^3$.

Within our context, we have to take into account two major constraints:

- voxels are not bound to be cubic (sampling frequencies are highly dependent of the acquisition means and are not identical in the general case – for instance the ratio between different axes may vary between 1 and 5 in confocal microscopy),
- the reduction factor of the re-sampling must be coherent to the surface representation.

That is why the previous formulation (15) is not usable as it is.

Making our voxel space isotropic in order to apply convolution operators coherently would be very memory expensive (the resolution would become the lowest common multiple of the sampling frequencies). Instead, by defining a real workspace corresponding to the discrete structure including the initial data, we will realize the convolution operations efficiently.

Our goal is to determine a list of volumic discrete images that we denote $G_0, G_1, \ldots, G_{max}$ and which represents the three-dimensional pyramid. G_0 is the initial image (given for segmentation) of sizes M, N and P. It is the image that possesses the greatest amount of informations. G_{max} will be the image that includes only the lowest frequencies. Let M_h, N_h and P_h be the sizes of the discrete image G_h for h between 0 and max. Their values are still unknown. Let I_h be the Cartesian space $M_h \times N_h \times P_h$. With these definitions a discrete image G_h is a function from I_h toward $[0, 1]$.

We denote I_R the space defined by the real image of size $[0, \mu[\times[0, \nu[\times[0, \pi[$. Because all images G_h represent at different scale the same real image, all of them have a real size of μ, ν, π. The immersion of a voxel (i, j, k) of a discrete image G_h into the real image space I_R is given by the transformation $\mathcal{T}_h$ (depending on the level of the pyramid) as below:

$$\begin{array}{rccc} \mathcal{T}_h & : & I_h & \longrightarrow & I_R \\ & & (i,j,k) & \longrightarrow & \left(i\frac{\mu}{M_h}, j\frac{\nu}{N_h}, k\frac{\pi}{P_h}\right) \end{array} \tag{16}$$

It is so an immersion preserving the real proportions of a 3-D discrete image in the parallelepiped defined by its real image.

We call *unit* of the real space and we denote it U_h the value $\min(\mu/M_h, \nu/N_h, \pi/P_h)$. It's the smallest distance between the immersions of two voxels in the real image (see **section 5.2**). In the case of an isotropic image, we got $U_h = \mu/M_h = \nu/N_h = \pi/P_h$.

In order to build the successive pyramid levels, we need to know the reduction factor. Being for the moment unpredictable (see **section 5.1**), our pyramid construction must authorize any reduction factor. Unlike purely discrete formulations, the transformation into a continuous image (see **section 4.1**) associated with our immersion process allows us to build pyramids of any factor. We can nevertheless notice that **[19]** has adapted the building mechanism of discrete pyramids to rational reduction factors. However the so-defined transformation is not a convolution process; thus the filters are not low-pass ones and the resulting signals are not well defined.

Another constraint consists in respecting the coherence of the filtering/re-sampling operation: we must verify that the filtering of high frequencies is in harmony with the reduction factor value. We have chosen to keep a convolution kernel of side 5. Therefore the reduction factor per dimension, denoted T, must be less than two.

Let V_0 be the base of our pyramid of real three-dimensional images. V_0 is given by the immersion then by the interpolation of the discrete data. As a matter of fact $V_0 = \Pi_{G_0}$. Let V_h be the level h of the real image pyramid. V_{h+1} is calculated from V_h. The number and the localization (in I_R) of the points to be calculated are determined by the reduction factor T, and the values are obtained after convolution of some points of V_h. Their storing after computation on the real image space is of course done in an array of voxels, assimilated to the discrete pyramid (G_i) at level $h + 1$.

The discrete sizes M_h, N_h, P_h and the measure unit U_h correspond to a real image V_h. Its real sizes (μ, ν, π) are of course constant for all h. Its characteristics are defined recursively with:

$$\begin{array}{cccc} M_0 = M & N_0 = N & P_0 = P & U_0 = \min(\mu/M, \nu/N, \pi/P) \\ M_{h+1} = \lfloor \frac{M_h}{T} \rfloor & N_{h+1} = \lfloor \frac{N_h}{T} \rfloor & P_{h+1} = \lfloor \frac{P_h}{T} \rfloor & U_{h+1} = U_h \cdot T \end{array} \tag{17}$$

Let $R(i', j', k')$ be a voxel of the discrete data of G_{h+1}. Our goal is to find its value for any (i', j', k').

Its immersion R_V in the real image V_{h+1} has coordinates of $\mathcal{T}_{h+1}(i', j', k')$ alias $\left(i\frac{\mu}{M_h}, j\frac{\nu}{N_h}, k\frac{\pi}{P_h}\right)$ (see **figure 11a**).

In order to establish the value of R, the convolution operation is defined over points of V_h. The central point has the same position in V_h and in V_{h+1}. The localization of the other points involved in the convolution ($5^3 - 1$ in 3-D for a kernel of size 5) is determined via the use of the unit U_h to discretize V_h around the point R_V (see **figure 11b**). So we obtain the convolution formula below:

Supposing G_h is known, V_h is then defined by Π_{G_h} and we got

$$G_{h+1}(i', j', k') = \sum_{m=-2}^{2} \sum_{n=-2}^{2} \sum_{p=-2}^{2} \omega(m, n, p) \cdot V_h(\mathcal{T}_{h+1}(i', j', k') + (mU_h, nU_h, pU_h)) \tag{18}$$

G_{h+1} defined then V_{h+1} implicitly ($V_{h+1} = \Pi_{G_{h+1}}$).

Because of the unknown reduction factor, the 5^3 points involved in the convolution doesn't coincide with given points of G_h in the general case (see **figure 11c**). Moreover there usually won't be any cover between points involved in two neighbouring convolutions.

Besides, each point of V_h compulsory for the convolution computation is interpolated from 8 data points of V_h (so stored in G_h) which form the parallelepiped containing this point (see (14) in **section 4.1** and **figure 11c** too).

The Gaussian convolution kernel (of size 5^3) is applied successively along the three dimensions because of its separable property. We can estimate the optimization savings by using the following notations and equations: Let T be the reduction factor, 5 the size of one side of the Gaussian kernel, t' the access time to a point value, t_1 the execution time of a classical algorithm, t_2 the execution time of the optimized algorithm. We got:

$$\frac{t_1}{t'} = 5^3 M_{h+1} N_{h+1} P_{h+1} \qquad \frac{t_2}{t'} = 5M_{h+1}N_hP_h + 5M_{h+1}N_{h+1}P_h + 5M_{h+1}N_{h+1}P_{h+1}$$

$$\text{Hence} \quad \frac{t_2}{t_1} = \frac{1}{25}(T^2 + T + 1) \tag{19}$$

The optimized algorithm is thus faster when the reduction factor T is between 0 and $\frac{-1+\sqrt{97}}{2}$ (approximately 4.42). An overview of the 3-D pyramid building algorithm is given in **appendix A**.

5 Segmentation

5.1 Image/Model Appropriateness

In **section 4.2.1** we have chosen to express surface-image interaction with constraints locally computed around each vertex. The use of a pyramid of three-dimensional images requires a model

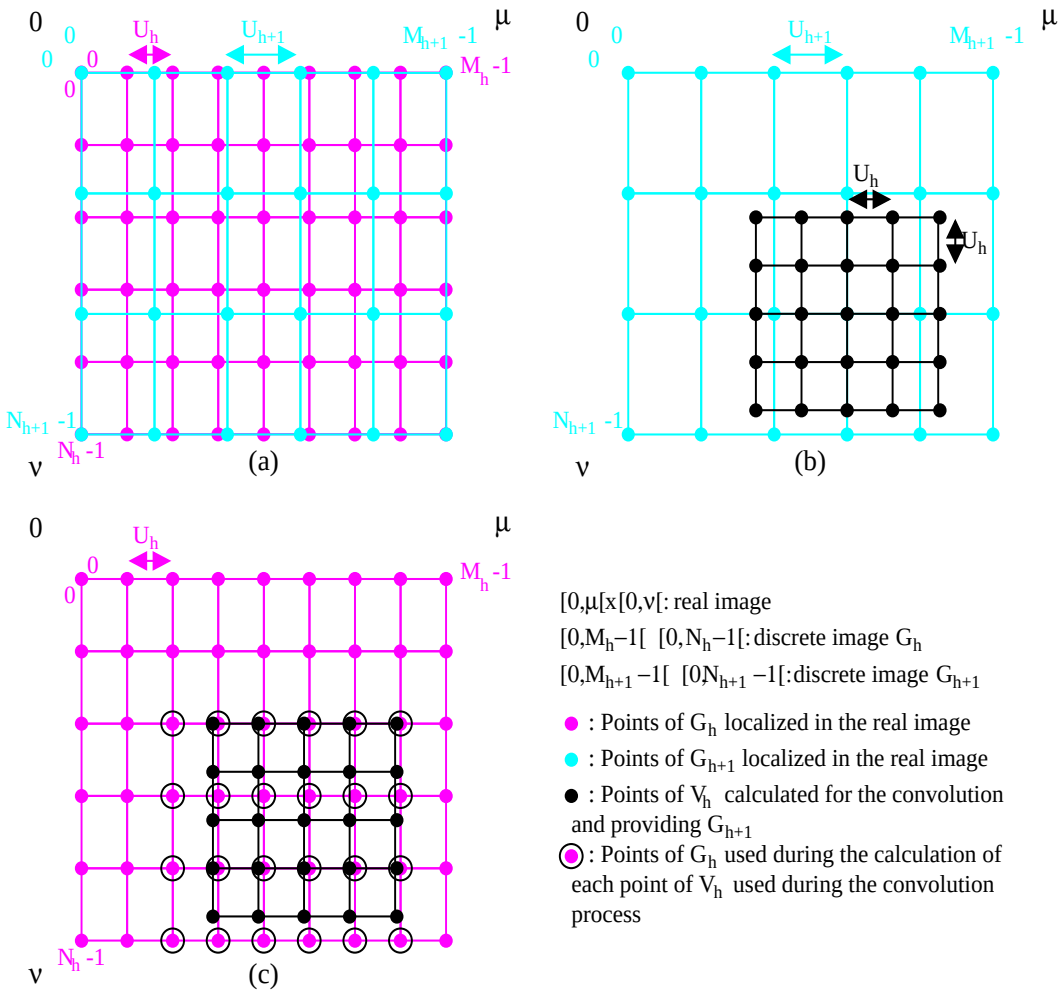


Figure 11: View of a convolution computation in 2-D: (a) The two superposed levels G_h and G_{h+1}, (b) computation of level G_{h+1} together with the localization of the convolution mask applied to one point, (c) application of the convolution mask over level G_h and visualization of the discrete points of G_h involved in the computation.

sufficiently flexible to adapt its refinement to image precision. The edges of our model should neither be too long, otherwise the high frequency contours could be missed, nor too small, because it would then represent just the decomposition of a small contour of two voxels. In order to obtain a correct appropriateness between the surface and the pyramid images, we must first examine the relations linking the model precision to the resolution of an image, and secondly, we must establish an algorithm of surface refinement, which ensures the appropriateness surface-image during the whole coarse-to-fine process.

5.1.1 Surface-image relationships

Section 3.3.1 has introduced a geometrical invariant δ which bounds the edge lengths. It is especially useful to simplify the solution of crossing problems in deformable polyhedrizations. According to equations (5) and (6), let d_{min} be the minimal edge length, d_{max} be the maximal edge length. We have $d_{min} = \delta$ and $d_{max} = 2.5\ \delta$.

The refinement of a triangulated surface can so be defined entirely with the invariant δ. Let δ_h be the invariant δ at level h of the pyramid. d^h_{min} and d^h_{max} are defined as same.

The image resolution is closely linked to the unit U_h previously defined in **section 4.3**. We take an interest to the case where images are isotropic (i. e. resolutions are identical along all three axes), the other case being solved in **section 5.2**. We recall that U_h is the real size of a voxel edge (cubic for an isotropic image) at level h of the pyramid. We can deduce the relationships between δ_h and U_h with the help of the following considerations:

1. According to **figure 12**, an edge can represent a contour formed by two 6-connected voxels (distant of U_h), so $d^h_{min} \leq U_h$,

2. According to **figure 12**, an edge can represent a contour formed by two stricly 26-connected voxels (distant of $\sqrt{3}\,U_h$), so $d^h_{max} \geq \sqrt{3}\,U_h$.

$$\text{Hence, } \frac{\sqrt{3}}{2.5} \leq \frac{U_h}{\delta_h} \leq 1 \quad \text{with } U_h = \min\left(\frac{\mu}{M_h}, \frac{\nu}{N_h}, \frac{\pi}{P_h}\right) \tag{20}$$

Whatever the value of U_h, an invariant δ_h which satisfies (20) may thus be found.

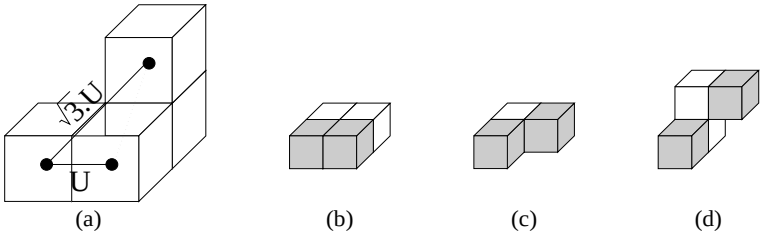


Figure 12: Appropriateness between edge length and voxel size: (a) ideal edge length, (b) example of 6-connected contour, (c) example of 18-connected contour, (d) example of 26-connected contour.

5.1.2 Surface refinement

Unfortunately a surface of given invariant δ may be built only during its initialization. After that, all modifications of this invariant are bound to geometrical constraints.

Now, the surface works in an image pyramid and, consequently, must refine its precision every time it goes down a level of the pyramid (see **figure 13**). We propose a process refining triangulated surface with a factor K. This factor will determine the reduction factor T of the pyramid.

Refinement process of the triangulated surface (or global surface division) (see **figure 14**):

1. in a first pass, a new vertex is created in the center of each facet of the model; the vertex is connected to the three vertices that delimit its facet,

2. in a second pass, the edges, which link together the old vertices (those which were not created during the first pass), are reversed in order to regularize edge lengths in a systematic way.

Such an algorithm reduces the average edge length to $1/\sqrt{3}$ of the old one. We may so apply to the invariant a reduction factor whose value is also $\sqrt{3}$, so $K = \sqrt{3}$. In order that the inegality (20) be respected at the initialization moment and during all successive pyramid levels, an identical reduction factor is chosen for the pyramid construction:

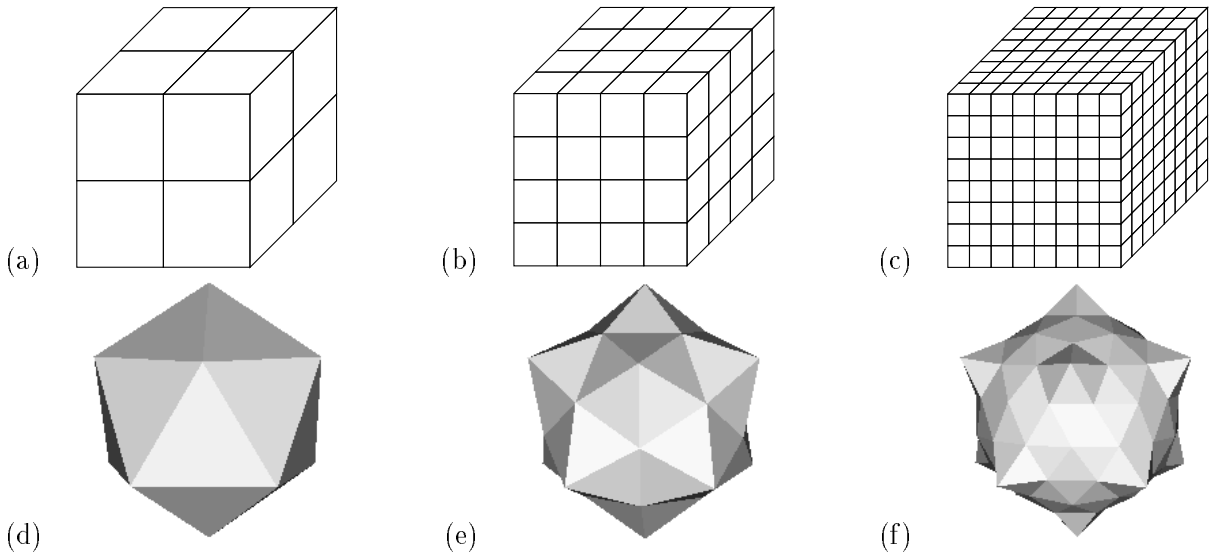


Figure 13: Resolution of a pyramidal image and precision of a triangulated mesh: (d) evolves in (a), (e) in (b), (f) in (c).

$$T = k = \sqrt{3} \quad \text{and} \quad \forall (h = 0 \ldots h_{max} - 1), \begin{cases} \delta_{h_{max}} = \delta_{init}, \text{ and } \delta_h = \delta_{h+1}/K \\ U_0 = U, \text{ and } U_{h+1} = U_h \cdot T \\ \text{we got thus } \frac{\sqrt{3}}{2.5} \leq \frac{U_h}{\delta_h} \leq 1 \end{cases} \tag{21}$$

At the initialization moment, a bubble or a set of bubbles, whose invariant δ_{init} is consistent with (20) at level h_{max}, are created. After that, the recursive process described by (21) guarantees a correct surface-image appropriateness, whichever are the iteration or the current level in the pyramid.

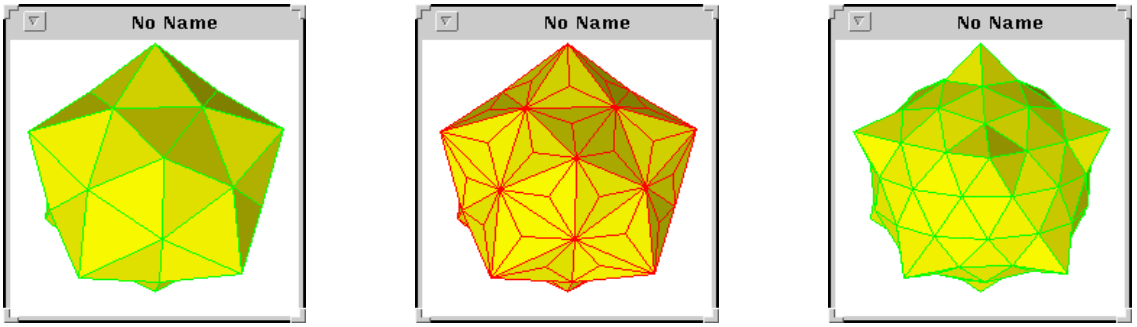


Figure 14: Example of a global division process over a polyhedron with sixty facets: (a) before global division, (b) after first pass, (c) after second pass.

5.2 Treatment of image anisotropy

We have to solve differenly image anisotropy according to the computation step, i. e. during the pyramid construction and during the image segmentation of a pyramid level.

5.2.1 Anisotropy during the pyramid construction

It is compulsory that the convolution mask applied during the construction be isotropic with respect to the real space where the image is immersed. If this is not properly done, pyramids will tend to maintain the contours following a direction where image resolution is fine, and to soften too much those following a direction where image resolution is proportionnaly coarse. That is why we have introduced in **section 4.3** the unit U_h, h indicating the level of the pyramid.

Within the context of anisotropic images, the applied filter is therefore defined from the unit $U_h = \min(\mu/M_h, \nu/N_h, \pi/P_h)$ and we realized the convolution process as described in **section 4.3**.

5.2.2 Anisotropy during segmentation

During the segmentation process, the edges have to keep their meaning with regards to the voxel space. On one hand, if we decide to work in the real image with sizes (μ, ν, π), edges loose their coherence with respect to the resolution and the precision of the informations. On the other hand, working in a real image where voxels are cubic, modifies the constraints that must be applied; the physical modelization looses then some of its realism. So we have drawn three different ways to deal with this problem:

- The surface evolves in a real space of sizes (μ, ν, π) and respects the constraints of the physical modelization. The surface-voxel appropriateness is therefore guaranteed only on the fine resolution axes.
- The surface evolves in a real space derived from the space (μ, ν, π) by affine transformation. This space has the same proportions than the discrete image it interpolates and its sizes are thus $(\lambda \cdot M_h, \lambda \cdot N_h, \lambda \cdot P_h)$. The internal forces have a slightly different behaviour than the one they would have in the real physical space.
- The surface evolves in a real space of sizes (μ, ν, π) which we equip with an anisotropic metrics. This metric is defined from the current discrete image (M_h, N_h, P_h) and maintains a constant triangulated mesh-voxel coherence along all axes while following the physical modelization.

The first method gives good results when the anisotropy is weak; the second one provides good results even if the anisotropy is more relevant; the last one, harder to implement (especially for the vectors operations), is a suitable solution whichever is the context. However, this last method is far from being essential within the segmentation framework, where the physical modelization is not supposed to simulate a very fine behaviour and is just a support to guide the surface. An implementation of these methods is presented in **section 6**.

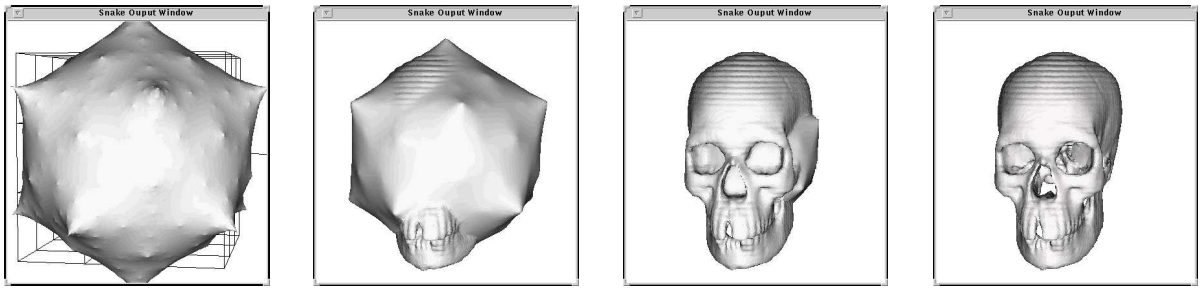
6 Results

We first test our model on a volumic image of a human skull [1] of discrete sizes $256 \times 256 \times 68$ and of real sizes $1.0 \times 1.0 \times 1.0625$. This image being highly anisotropic, we use the second method described in **section 5.2.2** in order to solve this problem. We set the physical parameters to the following values: $\lambda_c = 0.05$ and $\lambda_e = 0.0$ for the internal forces, $\lambda_i = -1.0$, $\Pi_0 = 0.08$, $\lambda_{di} = 0.0$ and $\mu_{di} = 0.0$ for the external forces. The segmentation algorithm with pyramid is described in **appendix B**. Few forces are needed because we neither wish to track the gradient maxima nor wish to obtain a regularly sampled triangulated mesh.

[1] Thanks to Yves Usson (TIMC-IMAG) for the volumic database.

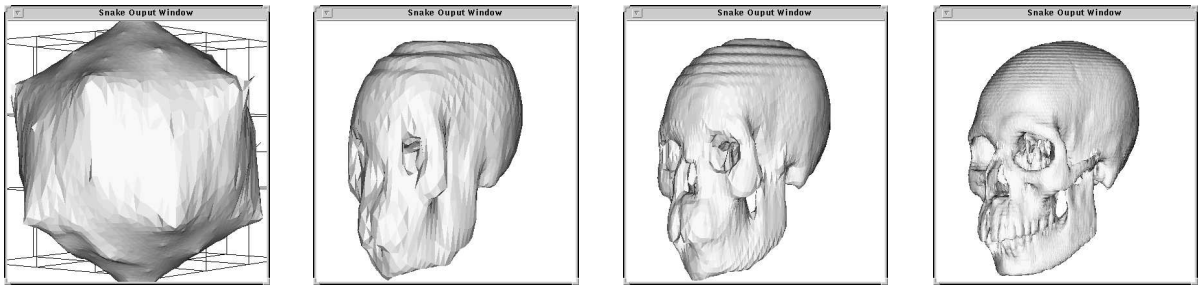
Moreover we provide our process with a full reliable heuristic that quickens the treatment of motionless or quasi-motionless vertices. The segmentation process is run two times for comparison purposes:

- We first run the process directly on the volumic image without using a pyramid. **Figure 15** shows surface evolution during its deformations: the surface slowly sticks on the outer part of the skull and then goes inside to segment its inner part (orbits of the eyes, brain cavity, ...). More than 400 iterations are necessary for the surface to stick perfectly on the inner part of the skull. The image possessing a very fine resolution, the initialization with an adequate triangulated mesh leads to a too costly segmentation in terms of computational time; so the model just segments the corresponding image of level one in the pyramid. That fact justifies all the more the use of pyramids as the ones we proposed.
- We run after the process by making use this time of the pyramid built up from this volumic image with a reduction factor of $\sqrt{3}$. The process waits for its complete convergence at one pyramid level before going down one level. **Figure 16** represents surface evolution in an image pyramid. The surface, at first coarse, looks quickly like the wanted shape and relies on the segmentation of one level to outline the object on next level.



(a) (b) (c) (d)

Figure 15: Surface evolution during a segmentation without pyramid: (a) iteration 0 on image G_1, (b) iteration 100 on image G_1, (c) iteration 250 on image G_1, (d) iteration 700 on image G_1.



(a) (b) (c) (d)

Figure 16: Surface evolution during a pyramidal segmentation: (a) iteration 0 on image G_3, (b) iteration 225 on image G_3, (c) iteration 333 on image G_2, (d) iteration 461 on image G_1.

Figure 17 analyzes the behaviour of both algorithms. The one of the classical segmentation algorithm is quite simple: the kinetic energy curve shows the slow segmentation convergence (see **figure 17b**), the vertex number (see **figure 17d**) and the average edge length (see **figure 17c**)

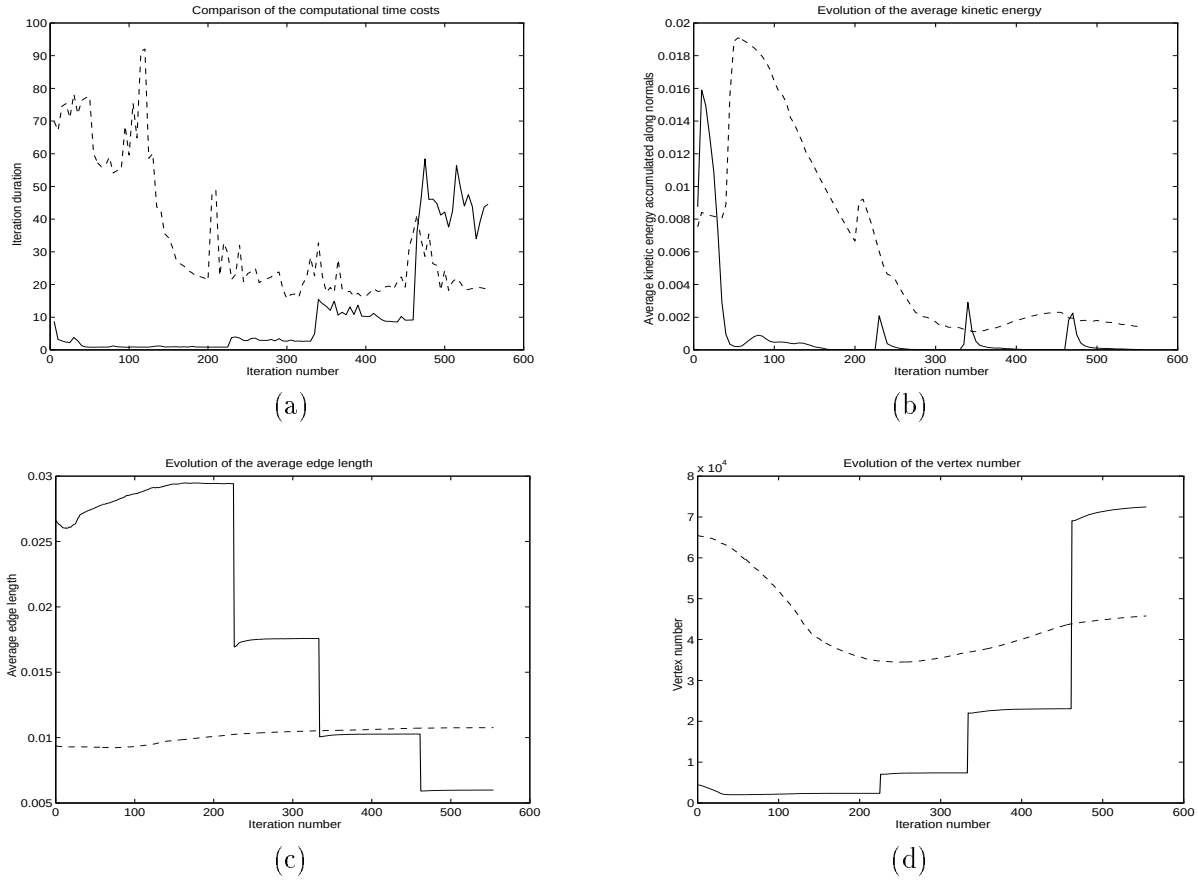
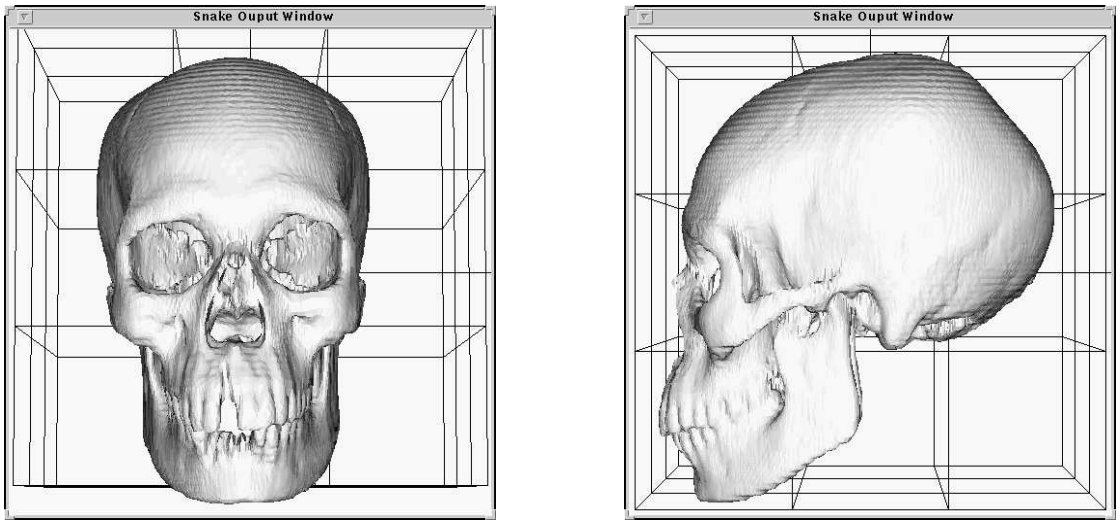


Figure 17: Statistics over a skull segmentation process and comparison between the approach without a pyramid (in dotted line) and the pyramidal approach (in solid line): (a) duration of each iteration, (b) evolution of the average kinetic energy accumulated along the surface normals, (c) average edge length according to the iteration, (d) vertex number of the surface according to the iteration.

are subject to few changes, the time cost (see **figure 17a**) slowly decreases but only because of the use of the heuristic. The behaviour of the pyramidal segmentation algorithm displays the four used pyramid levels: the vertex number (see **figure 17d**) and the average edge length (see **figure 17c**) show that the triangulated mesh has gone down a level at the iterations 226, 334 and 462. The kinetic energy evolution (see **figure 17b**) highlights the convergences at each step and the duration graph of each iteration (see **figure 17a**) explains the time savings provided by a coarse-to-fine process.

Figure 18 and **figure 19** show some different views of the totally segmented skull. The triangulated surface has one connected components and twenty-three topological holes.



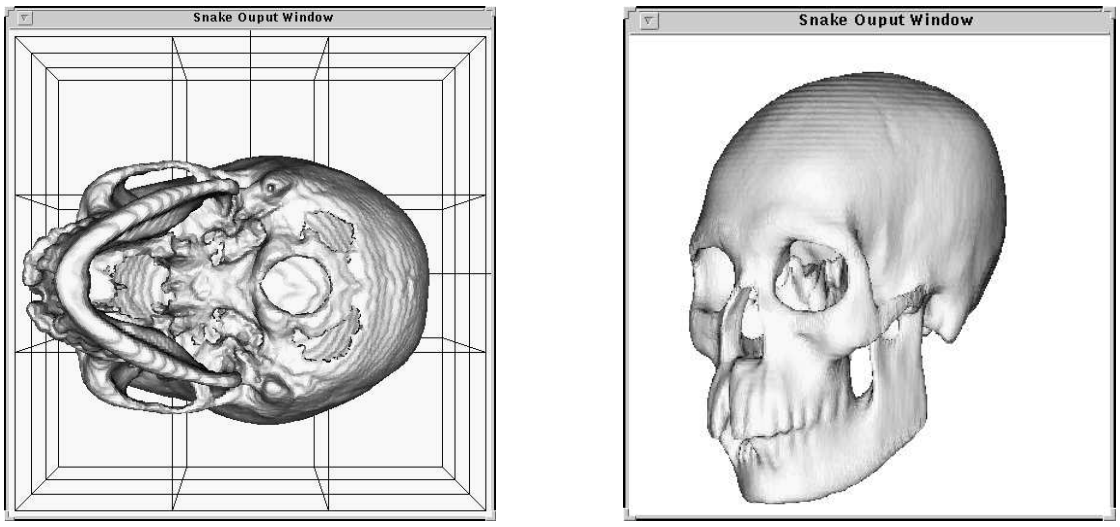
(a) (b)

Figure 18: Final results of the segmentation of G_0 (at the 560^{th} iteration the surface possesses 82473 vertices, 247551 edges and 165034 facets): (a) front view, (b) side view.

We then test our model on a more problematic image: a phase contrast MR angiographic image [2]. Its discrete sizes are $256 \times 256 \times 124$. We can observe that such images are mainly composed of vessels whose proportions are not suited for a pyramidal representation. Within this context, the top image of the pyramid represents nearly nothing (there are few low frequency informations) and the initialization of the pyramid is very bad. Even in this case, the model succeeded in following the vessels to recover the forgotten ones as shown by **figure 20**, but the convergences at each pyramid level are slower than in the first example. The physical parameters were set to the following values: $\lambda_c = 0.07$ and $\lambda_e = 0.0$ for the internal forces, $\lambda_i = -1.0$, $\Pi_0 = 0.05$, $\lambda_{di} = 0.0$ and $\mu_{di} = 0.0$ for the external forces.

We eventually test our model on a synthetic fractal image (the classical Sierpinski's cube). Its discrete sizes are $81 \times 81 \times 81$. We can observe that such images involve a lot of topology breaks. It is a good example to test the pyramid approach and the model has a correct behaviour, even if the great amount of topological problem slows the process (see **figure 21**). The physical parameters were set to the following values: $\lambda_c = 0.05$ and $\lambda_e = 0.001$ for the internal forces, $\lambda_i = -1.0$, $\Pi_0 = 0.4$, $\lambda_{di} = 0.0$ and $\mu_{di} = 0.0$ for the external forces.

[2] Acknowledgements to the UMDS Image Processing Group, London, for the angiographic image.



(a) (b)

Figure 19: Final results of the segmentation of G_0 (at the 560^{th} iteration the surface possesses 82473 vertices, 247551 edges and 165034 facets): (a) bottom view, (b) three quarters view of the same surface after a smoothing obtained with a modification of the curvature resistance parameter ($\lambda_c = 0.3$).

7 Conclusion

We have designed and developed an efficient volumic segmentation algorithm by means of a deformable triangulated surface evolving in a three-dimensional image pyramid. The obtained results show that their quality is identical to other similar algorithms; the tests, as for them, demonstrate the efficiency and the quickness of our algorithm.

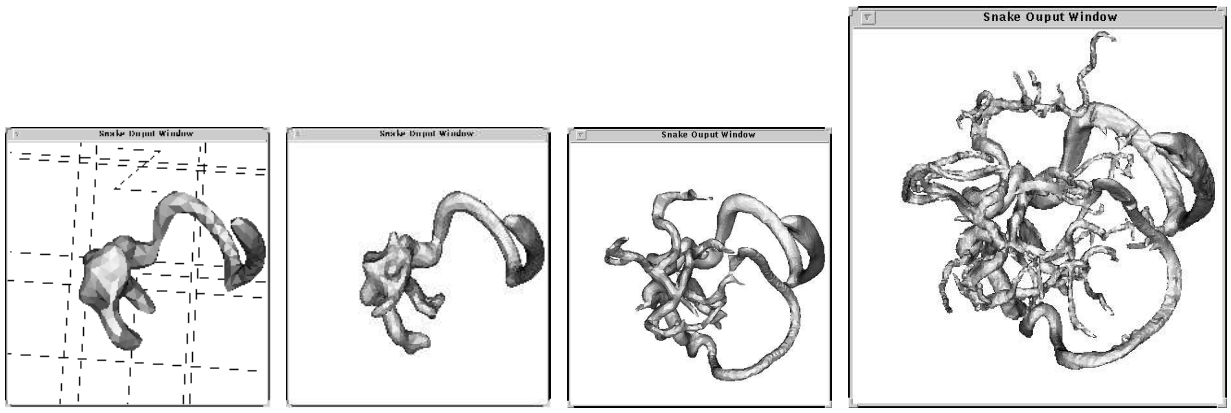
Several points may be nevertheless explored:

- The first convergence step can be greatly speeded up if the surface initialization contains more information. For instance, it would be very interesting to recover the surface coming from a Marching Cube process applied on the highest level of the pyramid ([**14**] and [**16**] for a parallelization). The surface computation, very fast because of the very coarse resolution, provides a first approximation of the object to segment. The algorithm benefits from a quicker convergence on the first level while avoiding the main problem set by marching-cube algorithm: the noise sensibility. However the coherence problem of the so-computed surface is still to be solved. The marching-cube algorithm indeed doesn't ensure at all the surface closing. Several enhanced versions are under consideration [**20**].

- The convergence may be also guided by some new constraints, for instance by introducing attractive forces generated either by *sure* points of the object or by particular edges detected during a pre-treatment.

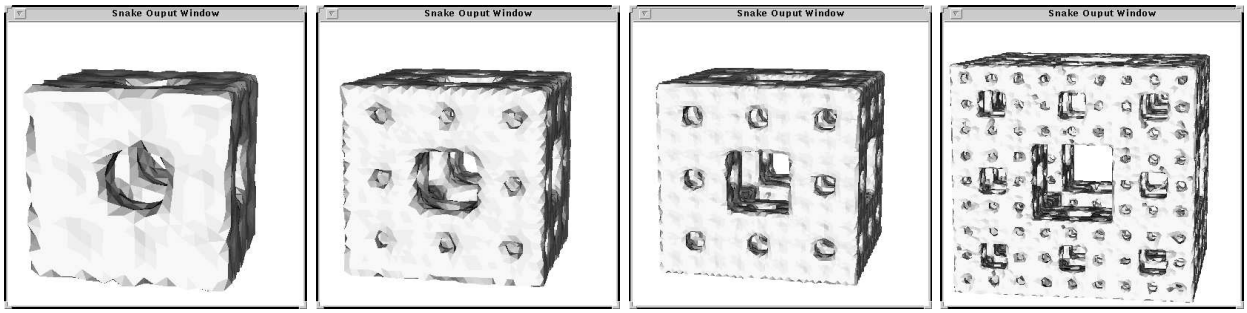
- In order to widen the application scope of our model, one can complete the physical formulation by adding global physical parameters, such as global speed or instantaneous rotation vector. Local components must of course be preserved in order to represent deformations.

- The overall speed can be improved by accelerating the detection of topological breaks. For instance [**24**] proposes an efficient algorithm for detecting self-collision. Therefore a more



(a) (b) (c) (d)

Figure 20: Surface evolution during the segmentation of an angiographic image with pyramid: (a) After convergence on image G_3, (b) After convergence on image G_2,(c) After convergence on image G_1, (d) Final result



(a) (b) (c) (d)

Figure 21: Surface evolution during the segmentation of a synthetic image with pyramid (Fractal volume of Sierpinski: (a) After convergence on image G_3, (b) After convergence on image G_2,(c) After convergence on image G_1, (d) Final result

constraining organization into hierarchy would reduce the cost of self-collision detection, which is for the moment within $\mathcal{O}(n \cdot \log(n))$ if n is the vertex number.

These proposed extensions are as many domains to explore.

References

[1] E. Bardinet, L. D. Cohen, and N. Ayache. "Fitting 3D Data Using Superquadrics and Free-Form Deformations". In *12th International Conference on Pattern Recognition*, volume 1, pages 79–83. IAPR, October 1994.

[2] P. J. Burt. "Fast filter transforms for image processing". *Computer Graphics and Image Processing*, 16:20–51, January 1981.

[3] P. J. Burt and E. H. Adelson. "The Laplacian pyramid as a compact image code". *IEEE Transactions on Communications*, 31:532–540, April 1983.

[4] A. Chéhikian. "Algorithmes optimaux pour la génération de pyramides passes-bas et laplaciennes". *Traitement du Signal*, 9:297–308, January 1992.

[5] L.D. Cohen. "Note on active contour models and balloons". *CVGIP*, 53(2):211–218, March 1991.

[6] H.B. Griffiths. *"Surfaces"*. Cambridge University Press, January 1976.

[7] J. M. Jolion and A. Rosenfeld. *"A pyramid framework for early vision"*. Kluwer, January 1994.

[8] M. Kass, A. Witkin, and D. Terzopoulos. "Snakes: active contour models". In *1st Conference on Computer Vision*, Londres, June 1987.

[9] S. Kumar and D. Goldgof. "A Robust Technique for the Estimation of the Deformable Hyperquadrics from Images". In *12th International Conference on Pattern Recognition*, volume 1, pages 74–78. IAPR, October 1994.

[10] J-O. Lachaud and E. Bainville. "A discrete adaptative model following topological modifications of volumes". In *4th Discrete Geometry for Computer Imagery*, pages 183–194, September 1994.

[11] J.O. Lachaud and A. Montanvert. "Segmentation tridimensionnelle hiérarchique par triangulation de surface". In *10ème Congrès Reconnaissance des Formes et Intelligence Artificielle*, January 1996.

[12] F. Leitner and P. Cinquin. "Complex topology 3D objects segmentation". In *Advances in Intelligent Robotics Systems*, volume 1609 of *SPIE*, Boston, November 1991.

[13] F. Leitner, I. Marque, and P. Cinquin. "Dynamic Segmentation: finding the edge with snake-splines". In *Curves and Surfaces*, pages 279–284. Academic Press, January 1990.

[14] W. E. Lorensen and H. E. Cline. "Marching Cubes: A High Resolution 3D Surface Construction Algorithm". *Computer Graphics*, 21:163–169, January 1987.

[15] R. Malladi, J. A. Sethian, and B. C. Vemuri. "Shape Modelling with Front Propagation: A Level Set Approach". *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(2):158–174, February 1995.

[16] S. Miguet and J-M. Nicod. "A load-balanced parallel implementation of the Marching-Cube algorithm". In *HPCS*, July 1995.

[17] J.V. Miller, D.E. Breen, W.E. Lorensen, R.M. O'Barnes, and M.J. Wozny. "Geometrically deformed models: A method for extracting closed geometric models from volume data". *Computer Graphics*, 25(4), July 1991.

[18] O. Monga and S. Benayoun. "Using differential geometry in R4 to extract typical features in 3D density images". In *11th International Conference on Pattern Recognition*, volume 1, pages 379–382. IAPR, September 1992.

[19] S. Peleg and O. Federbusch. "Custom Made Pyramids". *Pyramidal Systems for Computer Vision*, F 25, January 1986.

[20] S. Röll, A. Haase, and Kienlin M. von. "Fast Generation of Leakproof Surfaces from Well-Defined Objects by a Modified Marching Cubes Algorithm". *Computer Graphics Forum*, 14(2):127–138, January 1995.

[21] R. Szeliski and D. Tonnesen. "Surface Modeling with oriented Particle Systems". Technical Report CRL-91-14, DEC Cambridge Research Lab., December 1991.

[22] S. Tanimoto and T. Pavlidis. "A hierarchical data structure for picture processing". *Computer Graphics and Image Processing*, 4:104–119, June 1975.

[23] D. Terzopoulos and A. Witkin. "Deformable Models: Physically based models with rigid and deformable components". *IEEE Computer Graphics and Applications*, 8(6):41–51, November 1988.

[24] P. Volino and Thalmann N. Magnenat. "Efficient self-collision detection on smoothly discretized surface animations using geometrical shape regularity". In *Eurographics'94*, volume 13(3), September 1994.

[25] R.T. Whitaker. "Volumetric deformable models: active blobs". In *VBC*, volume 2359 of *SPIE*, pages 122–134, March 1994.

A Building of three-dimensional pyramids

Let I be the work-image of discrete sizes $M \times N \times P$ and of real sizes $\mu \times \nu \times \pi$.
We denote T the reduction factor and T_{max} the desired maximal reduction.

$h \leftarrow 0, G_0 \leftarrow I, M_0 \leftarrow M, N_0 \leftarrow N, P_0 \leftarrow P$.
While $(T^h < T_{max})$ do
    computation of $M_{h+1}, N_{h+1}, P_{h+1}$.
    convolution of V_h, derived from G_h along the first axis by the filter
        [1 4 6 4 1]/16; the result is stored in G'_h of sizes $M_{h+1} \times N_h \times P_h$.
    convolution of V'_h, derived from G'_h along the second axis by the filter
        [1 4 6 4 1]/16; the result is stored in G''_h of sizes $M_{h+1} \times N_{h+1} \times P_h$.
    convolution of V''_h, derived from G''_h along the third axis by the filter
        [1 4 6 4 1]/16; the result is stored in G_{h+1} of sizes $M_{h+1} \times N_{h+1} \times P_{h+1}$.
    memory clearing of G'_h and G''_h.
    $h \leftarrow h + 1$.
End of while

B Volumic segmentation algorithm

$h \leftarrow h_{max}$ an $\epsilon > 0$ is defined by the user.
Initialization of a bubble with adequate refinement δ according to the image G_h.

Repeat
    Repeat
        Computation of normals and barycenters for each surface vertices.
        Vertex displacement according to internal forces and constraints defined
            by image G_h.
        Repeat
            Checking of the local geometric constraints and modification of
                the surface geometry accordingly.
            $rupture \leftarrow$ presence of a topological rupture.
            if $(rupture)$ then solution of the topological problem.
        Until $(rupture == false)$
        $E_c \leftarrow$ average kinetic energy of displacements normal to surface.
    Until $(E_c \leq \epsilon)$
    Surface global division, $\delta \leftarrow \delta/\sqrt{(3)}$.
    $h \leftarrow h - 1$
Until $(h < 0)$