Deformable Meshes with Automated Topology Changes for Coarse-to-fine 3D Surface Extraction

Jacques-Olivier Lachaud[1] and Annick Montanvert[2]

[1] Laboratoire Bordelais de Recherche en Informatique
Domaine Universitaire
351 Cours de la Libération
33405 Talence, France
e-mail: lachaud@labri.u-bordeaux.fr
phone: +33-(0)5.56.84.69.12

[2] Laboratoire des Images et des Signaux
961 rue de la Houille Blanche
Domaine Universitaire, BP 46
38402 Saint Martin d'Hères cedex, France
e-mail: annick.montanvert@inpg.fr
phone: +33-(0)4.76.82.64.67

September 19, 2000

Abstract

This work presents a generic deformable model for extracting objects from volumetric data with a coarse-to-fine approach. This model is based on a dynamic triangulated surface which alters its geometry according to internal and external constraints to perform shape recovery. A new framework for topology changes is proposed to extract complex objects: within this framework, the model dynamically adapts its topology to the geometry of its vertices according to simple distance constraints. In order to speed up the process, an algorithm of pyramid construction with any reduction factor transforms the image into a set of images with progressive resolutions. This organization into a hierarchy, combined with a model which can adapt its sampling to the resolution of the workspace, enables a fast estimation of the shapes included in the image. After that, the model searches for finer and finer details while relying successively on the different levels of the pyramid.

Key-words: 3D surface extraction, deformable model, adaptive topology, multi-resolution, 3D pyramid.

Running title: Coarse-to-fine 3D Surface Extraction

1 Introduction

Acquisition of 3D data, particularly in biomedicine, has become more and more usual this last decade. This kind of 3D data has the particularity to be "visually significant", as far as it can be perceived by a human observer.

Presentation of 2D data is trivial, but showing 3D data is a very difficult problem. In fact, several goals can be given: visualization and rendering, access to specific information (e.g., slices, grey levels), extraction of information (e.g., objects in 3D data, measurements of distances, areas, volumes). All these issues have been more or less studied for the past years.

Our own target is not to perform rendering: if rendering offers some possibilities to the human observer to understand the 3D data (for instance using ray tracing, opacity, colors, moving 3D data cube, etc), it does not provide any information which can be later used for a post-processing. As the same line, showing slices or selected grey levels does not provide further useful data.

Consequently, we are interested in extracting information necessary for automatic processing of 3D data. This could be: defining the objects, computing their position and size, recognizing them, etc.

Unlike 2D data, for which everyone can rather delineate objects on images, no tool can be provided to delineate 3D objects with a computer correctly. As a consequence, we base the extraction of objects from 3D data on the grey level information. A basic method for that is the *Marching-Cubes* algorithm [27], which extracts iso-surfaces for a given grey level, but many others exist (e.g., simplicial decomposition [13], digital surface tracking [14]). They provide a triangulated surface fitting a given grey level. With some improved algorithms, the computed surface is topologically consistent (i.e., closed and oriented). But iso-surfaces are too constrained, and we would like to obtain objects that can be at varying distances from iso-surfaces if "necessary". "Necessary" would mean that the surface can be somewhat "soft", more or less regularly sampled, and that other kinds of information than iso-potentials could be used (grey levels, similarity, gradients, etc).

To extract objects from 3D data, we propose a generic deformable model based on a triangulated mesh whose main characteristics are: shape recovery is performed via internal and external forces; no *a priori* is made on the topology of objects; the model automatically adapts its topology according to the geometry of its vertices; detection and resolution of topological breaks are optimized by regularizing edge lengths over the whole mesh; the mesh can be refined to be consistent with the image resolution; anisotropic images can be processed indifferently. With this lexible formulation, new constraints can easily be added (e.g. user interaction, forces derived from precomputed edge images).

Moreover, in order to provide a coarse-to-fine approach to the shape recovery problem, an algorithm of pyramid construction with any reduction factor is presented. The model can thus evolve in a pyramid of volumetric images to quickly outline shapes and progressively extract finer and finer details. The speed-up offered by the introduction of multi-resolution is exhibited and the robustness of this approach is tested.

In section 2, we recall related work on shape recovery by deformable models. In section 3, the physical and geometrical formulation of the model is presented; problems linked to variable and complex topology are also tackled in this section. Section 4 presents the construction of the pyramid of 3D data, the embedding of the model in this pyramid, and the coarse-to-fine process. In section 5, algorithms of shape recovery with or without multi-resolution are described, and their performance are compared; we test the robustness of the model on various databases, and we show several applications.

2 Related work

Deformable models are particularly suited to the problem of extracting objects from volumetric data. They are intensively used in segmentation [20], mapping [28], tracking and motion analysis [34] and non-rigid modeling [36]. Deformable models have various formulations :

Parametric form. Models based on quadric, superquadric [2, 38] and hyperquadric [21] segment images by computing both local and global parameters. In [38], a set of forces computed from the image and from model properties guides the surface towards the desired solution. In [2], several global parameters are computed with a least-square method

and then the result is sharpened by adjusting a control-box. In all cases, segmented objects must be homeomorphic to a sphere for correct results. The snake representation introduced in [20] is also a parametric model that segments images by energy minimization. Examples of three-dimensional extensions to snakes are parametric surfaces [7] ; in this approach, a finite element method is used to achieve minimization. The result has the same topology as the initial surface. To overcome this limitation, the surface snake can be embedded in the image grid [30] and the tessellation of the surface is updated after each step. In this extension, the model is no more parametric but rather related to deformable meshes.

Implicit form. Implicit models are widely used in the context of image synthesis. Some authors have proposed the use of implicit models in the context of MRI segmentation [40] and shape modeling [29]. Both methods use functions from $\mathbb{R}^4$ to $\mathbb{R}$ to represent variable topology in $\mathbb{R}^3$. Equations are solved by embedding the model in the image grid. A multi-resolution approach [40] or a restriction of the equation domain to a narrow band around the active shape [29] can be used to reduce the computational cost. Geometrical features can be extracted from a polygonal approximation of the implicit model.

Mesh form. Deformation is performed by constraining the model on its vertices [31]. Many deformable or adaptive meshes have been proposed. Some of them can handle objects with no *a priori* on their topology: simplex meshes as proposed in [8] are capable of representing any oriented two-manifold (even with boundaries); the user interacts to perform topological breaks. The mesh is considered as a mass-spring system whose nodes are its vertices. Regularization constraints or user constraints are thus easily defined. Some authors have proposed models whose evolution is not governed by a dynamic. Cubic spline surfaces as proposed in [25] modify their topology by using a series of three-stage evolution (contraction on convex areas, then contraction on hyperbolic areas, and after contraction on concave areas). Bicubic spline surfaces presented in [17] are more focused on the computation cost than the previous model: an iterative Newton minimization method is used to extract objects from distance transform images; the surface parameters are decoupled to speed up the computation of each iteration; the drawback is that this model cannot recover complex shape.

In this article, we present a generic model for recovering shapes or regions and extracting surfaces from three-dimensional data, for computing geometrical and topological informations, and for visualization. The model is flexible enough to be used as a complementary tool for segmentation, tracking, or matching. These properties induce the following constraints on its formulation:

- The model is explicit because providing geometrical (area, volume, local curvature) and topological (Euler-Poincaré characteristic) information is of great interest in clinical applications.
- The model is locally and globally deformable to perform a wider range of tasks and therefore has a dynamic topology. It can represent every kind of closed and oriented two-manifold (i.e. oriented two-manifold without boundary). In this way, no *a priori* knowledge on the topology of the final shape is needed.
- The evolution of the model is determined by the constraints applied on vertices. Constraints are either internal (elasticity, rigidity) or external (interaction with images or with user-specified constraints).

3 Presentation of the deformable model

3.1 Choice of representation

Our model is based on a triangulated surface that may have several connected components. All components are closed and oriented. The data structure minimizes the memory space: each vertex has a set of parameters (coordinates, normals, speed) and an oriented list of its neighbors. Hence, all oriented combinatorial two-manifolds without boundary can be represented [16] (refer to [12] for a definition of combinatorial two-manifolds). Boundaries are not allowed because they introduce ambiguities in topological *accidents* and need special handling of their constraints. Moreover, boundaries in surfaces are useless to recover the boundary of a volumetric object (the boundary of a three-manifold is a two-manifold without boundary).

The model does not intersect itself in the Euclidean space: during its evolution, the embedding of the model always represents the boundary of a volumetric object. A set of simple geometrical constraints is applied on the model to optimize both detection and resolution of topological accidents. Therefore, the model can adapt its topology to the geometry of its surface without any interaction of the user.

The mesh is considered as a set of particles linked with spring forces [1, 11]. External constraints are applied on vertices. As a consequence, image resolution and mesh precision must be similar.

3.2 Geometry and topology

A model based on triangulated meshes offers some immediate advantages: simplicity, speed and fast rendering. To these points we can add the opportunity to extract features of the object, such as the area and the volume defined by the object, moments or topological information. Local curvature can be approximated [5], geodesic distances on such surface can also be computed [41]. Thus, triangulated meshes provide an efficient tool to shape analysis.

A major problem that arises in shape recovery or segmentation is the possible complex topology of objects in 3D data. As a matter of fact, parametric models are bound to their intrinsic topology. Implicit models are naturally able to dynamically modify their "topology" *in* $\mathbb{R}^3$ but this "topology" is never explicit. Currently, one classical approach is to initialize the deformable model with an estimation of the topology of the final shape [1, 10]; the model is then deformed to fit the object more precisely. These methods perform well in the recovery of unstructured 3D data but they are not suited to volumetric data. Some deformable models offer dynamic topological modifications with user interaction or validation [8], or by performing series of contraction [25].

As far as we know, only the dynamic model presented in [30] automatically modifies its topology with regards to its variable geometry. It uses a simplicial decomposition of the image to segment: a set of tetrahedra is thus partitioning the image. At each step, movements of every vertex are computed. The moved vertices belong to a new set of tetrahedra. By analogy with a flame propagation algorithm, the model keeps track of the "burnt" vertices of the simplicial grid, which the surface has already crossed. By combining this information to the new set of tetrahedra, the model extracts the "surface" at the next time step. The nodes of the "surface" are recomputed at each iteration with a method similar to a *Marching-Tetrahedra* [13]. This *reparameterization* performs topological transformations in an implicit way. However, the model is bound either to inflate everywhere or to deflate everywhere. Therefore, it is delicate to modify the parameters of the model during the evolution or to provide a flexible interaction to the user. Tracking of surfaces in spatiotemporal data is also difficult. Besides, the accuracy of the model is linked to the resolution of the simplicial grid: increasing the resolution by two thus requires eight times as many tetrahedra.

In this article, we propose a model that adapts its topology to the geometry of its vertices at each step, without any user-interaction or any grid embedded in the image; topological modifications are locally done. The method is based on global geometrical constraints applied to the vertices of the model (see [22] or [23] for a precise description).

We denote $\mathcal{T}$ the triangulated mesh, $S_{\mathcal{T}}$ the set of vertices of $\mathcal{T}$, s the cardinality of $S_{\mathcal{T}}$ (i.e., the number of vertices). If $U \in S_{\mathcal{T}}$, then $\mathcal{V}(U)$ is the set of the neighbors of U, $d(U)$ the number of neighbors, $(U_i)_{0 \leq i < d(U)}$ the oriented list of the neighbors of U, $\mathbf{u}$ the coordinates of U. We also denote $\#F$ the cardinality of a finite set F.

We introduce an invariant δ ($\delta > 0$) which is associated with the mesh. Three geometrical constraints are derived from this invariant. They determine the sampling of the mesh:

$$\forall (U,V) \in S_{\mathcal{T}} \times S_{\mathcal{T}} / \; V \in \mathcal{V}(U), \; \delta < \|\mathbf{u} - \mathbf{v}\| \tag{1}$$

$$\forall (U,V) \in S_{\mathcal{T}} \times S_{\mathcal{T}} / \; V \in \mathcal{V}(U), \; \|\mathbf{u} - \mathbf{v}\| < 2.5 \; \delta \tag{2}$$

$$\forall (U,V) \in S_{\mathcal{T}} \times S_{\mathcal{T}} / \; V \notin \mathcal{V}(U), \; \frac{2.5}{\sqrt{3}} \; \delta \leq \|\mathbf{u} - \mathbf{v}\| \tag{3}$$

Constraints (1) and (2) express the upper and lower bounds of one edge length. They force the triangulated surface to remain rather regularly sampled. The fact that constraint (3) is not satisfied for a couple of non-neighboring vertices expresses a collision between two distinct parts of the surface. This constraint is used to detect self-intersection.

The numerical constant of (2) ensures that constraint (1) is followed after the creation of a vertex on the problematic edge. The numerical constant of (3) is the longest distance between a vertex of the surface and the three vertices of a facet which the vertex is going through. We assume that movements of vertices are small and that the inversion operation is used whenever triangles are too elongated (see Figure 1a). Under these two hypotheses, correct collision detection is achieved by this constraint [1].

Computing distances between vertices is very fast for (1) and (2) and checking these constraints can be done in $O(s)$. Constraint (3) over the whole mesh is checked in $O(s \log(s))$ with a point octree. This point octree must be computed at each iteration. Note that the test of constraint (3) can be performed even faster — theoretically in $O(s)$ — with a discrete "image" gathering vertices at discrete positions; in this article, only the point octree algorithm has been implemented. We do not use algorithms for collision detection based on OBB-trees [15] because they are optimized for rigid meshes with motion. Besides, collision detections based on hierarchical mesh representation [39] need fixed topology and are not suited to models with dynamic topology.

3.3 Topological changes

The model changes its topology according to the classical Eulerian topological transformations of creation, deletion or inversion (see Figure 1 in case of violation of (1) or (2)). These transformations are performed in $O(1)$.

Non-Eulerian topological transformations of closed and oriented surfaces that evolve in the Euclidean space are described on Figure 2. During their evolution, these surfaces can mainly meet two different transformations (refer to [23] for further details): *axial transformations*, where

[1] If these two hypotheses are omitted, then the numerical constant of (3) should be set to $\sqrt{19}/2$ (provided the numerical constant of (2) is 2.5). With these constants, it can be shown that the combinatorial triangulated surface is "embedded" in $\mathbb{R}^3$ as a 2-manifold without boundary.

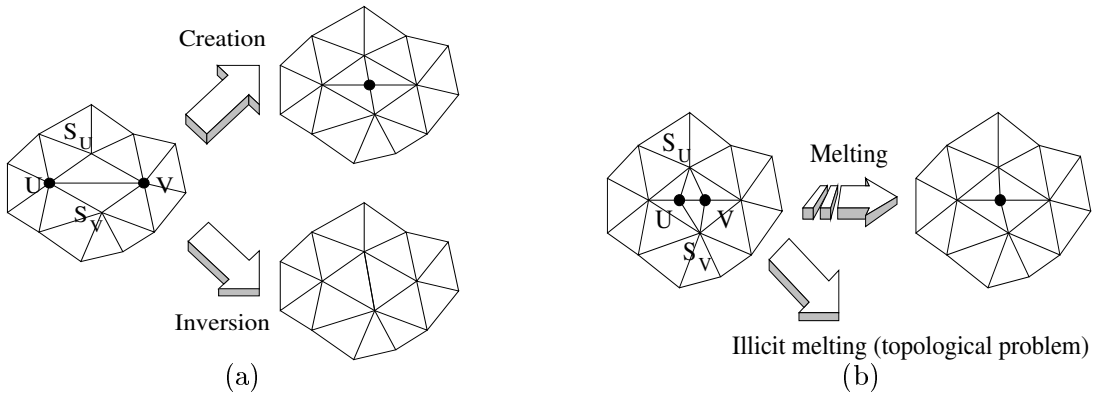


Figure 1: (a) A creation or an inversion is done if the couple (U, V) does not verify constraint (2) ; (b) A deletion is done or an annular problem is detected when constraint (1) is not satisfied by (U, V) .

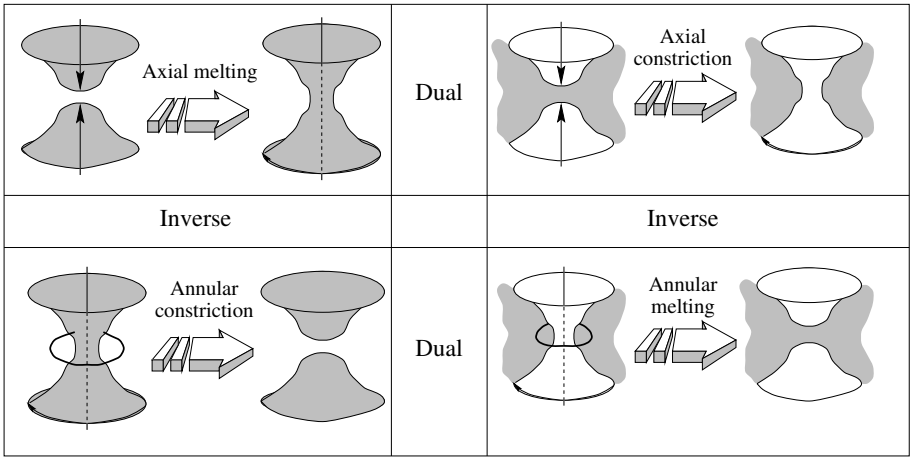


Figure 2: Description of the four main topological accidents for a closed oriented surface in $\mathbb{R}^3$

two parts of the surface which are not locally connected are colliding, and *annular transformations*, which occurs when a connected part of the surface homotopic to a circle is shrinking to a point.

Therefore, we have first to detect when a non-Eulerian transformation must take place, and secondly to exhibit the corresponding mesh operations.

Axial transformations. If equation (3) is not satisfied by a couple of non-neighboring vertices (U, V), then two different parts of the surface are colliding. Now, these two vertices may have common neighbors as it can be seen on Figure 3. Consequently, the fact that constraint (3) is not verified may hide an annular transformation. So intermediary vertices are created between U and its neighbors and between V and its neighbors (Figure 4a), and, only after, a triangulation is done between the neighbors of U and V (Figure 4b) (because of the intermediary vertices, U and its neighbors, and V and its neighbors, form two parts of the surface that are not locally connected).

Annular transformations. These transformations are detected with constraint (1). A tubular part of the surface is collapsing onto itself. If a couple of neighboring vertices does not satisfy constraint (1), then we have to check whether the neighborhood of these two vertices is a tubular part of the surface. To do so, the value $\#\left(\mathcal{V}(U) \cap \mathcal{V}(V)\right)$ is checked. If this value is equal to 2, then the two vertices U and V are simply merged into one vertex

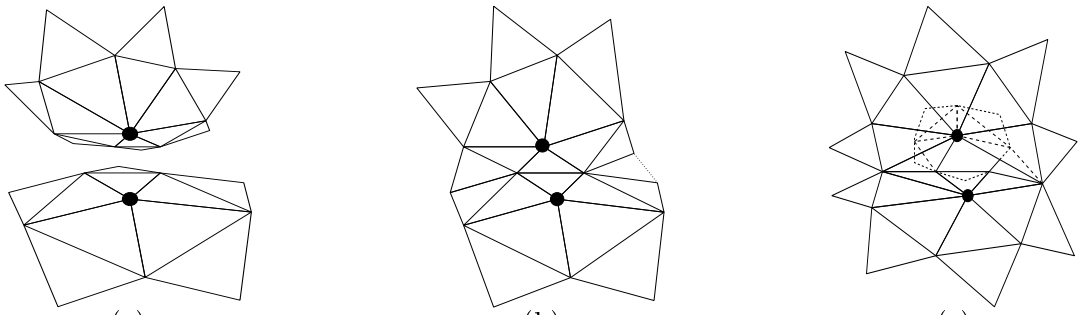


Figure 3: These figures show several configurations where the geometric constraint (3) is not satisfied : (a) no common neighbor; (b) one group of common neighbors; (c) two groups of common neighbors.

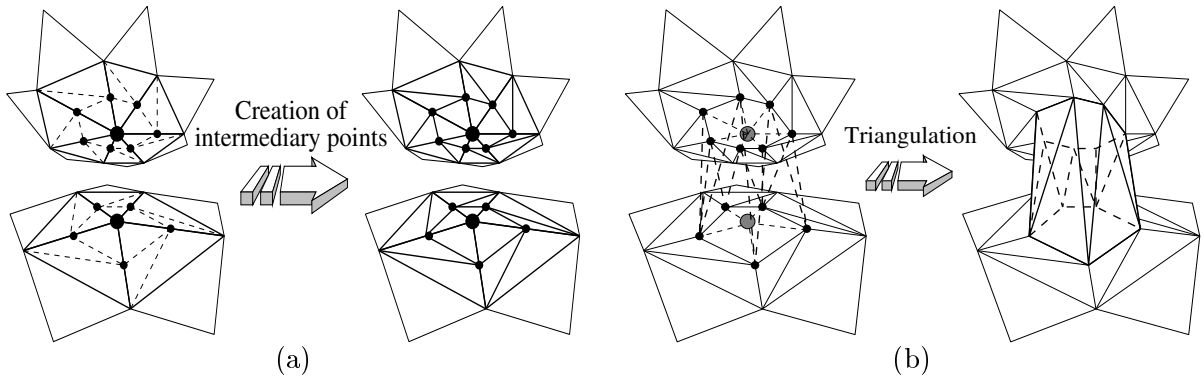


Figure 4: Implementation of axial transformations: (a) creation of intermediary vertices; (b) triangulation between the two surfaces and deletion of the two old vertices.

at their mid-point. Otherwise, the neighborhood of U and V is so bent that it forms a narrow tube (or possibly several narrow tubes joining up at this location). In this case, the surface is cut in two along the edges UV, VO, OU — O is a common neighbor of U and V, but OUV is *not* a face of the triangulated surface — (see Figure 5). After that, the two created holes are filled in by two triangles, and the split vertices (U_1V_1 and U_2V_2) can be merged separately.

It is easy to show that the so-built transformations follow the expected variations of the Euler-Poincaré characteristic. The proposed topological transformations obey both the evolution of the geometry of the vertices and the "embedding" — a more correct word would be *imbedding* — of the combinatorial surface as a two-manifold in $\mathbb{R}^3$ without boundary. Of course, there may be three or more parts of the surface which are not locally connected that are colliding. As well, a part of the surface which consists of several narrow tubes joining up at one place may collapse onto itself precisely at this location. In these cases — which are very uncommon but may theoretically occur —, the previous transformations are applied successively (annular transformations are performed before axial transformations).

All non-Eulerian transformations are performed in O(1). Deletion of a tetrahedron can also be done with the same performance. The creation of a tetrahedron is governed by the user.

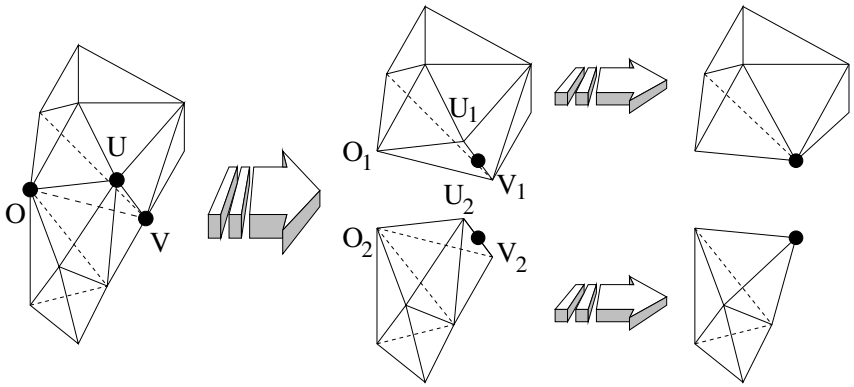
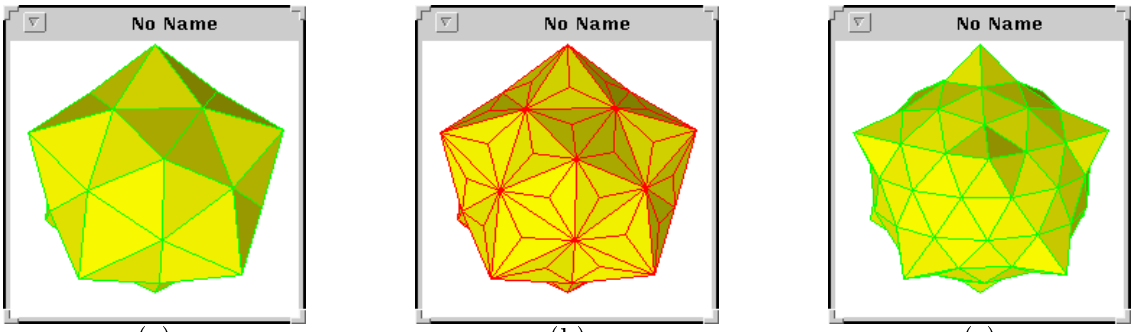


Figure 5: Implementation of annular transformations: splitting of the problematic set UVO in two parts, then separate merging.



(a) (b) (c)

Figure 6: Example of a global refinement over a polyhedron with sixty facets: (a) before refinement; (b) after first pass; (c) after second pass.

3.4 Global refinement

We propose an Eulerian transformation denoted $\Delta_{\frac{1}{\sqrt{3}}}$ that globally refines any triangulated mesh (see Figure 6). This transformation reduces the average edge length to $\frac{1}{\sqrt{3}}$ of the old one, thus increasing the sampling by $\sqrt{3}$. This transformation allows multi-resolution modeling. It can be decomposed into two stages:

1. In a first pass, a new vertex is created at the mid-point of each facet of the model; the vertex is connected to the three vertices that bound its facet.

2. In a second pass, the edges which connect the old vertices (those which were not created during the first pass) are inverted in order to regularize edge lengths.

This algorithm reduces the average edge length to $1/\sqrt{3}$ of the old one. The global associated invariant δ must also be divided by $\sqrt{3}$. The transformed surface possesses about three times as many vertices as before transformation. This refinement algorithm has been preferred to another one based on the creation of one vertex on each edge, because the former is adapted to the data structure of the mesh: this structure where every vertex has an ordered list of its neighboring vertices is consistent only when all face elements are triangular. This would be false during the refinement process if the algorithm created four triangles on each triangle.

3.5 Dynamic

The mesh is assimilated to a dynamic system of particles, which are the vertices of the mesh, following both constraints of other particles and of environment. Thus, interactions between particles occur only between direct neighbors and the computation time of internal constraints is in $O(s)$ for the whole mesh. We have chosen a dynamic evolution for the vertices of the model (vertices undergo inertia: mass m is greater than zero). This choice is motivated by our personal experience. We have indeed observed that the model has better convergence properties when mesh vertices undergo an inertia (in particular when internal forces are strong). Besides, the evolution of the mesh is then less sensitive to the parameters (especially to the damping factor).

Let X be a vertex of $S_{\mathcal{T}}$. We denote $\mathbf{x}$ its coordinates ($\dot{\mathbf{x}}$ its velocity and $\ddot{\mathbf{x}}$ its acceleration). The Newtonian law of motion is applied on each vertex:

$$m\ddot{\mathbf{x}} + \gamma\dot{\mathbf{x}} = \mathbf{f}_{int} + \mathbf{f}_{ext}, \tag{4}$$

where m is the mass of the vertex and γ the damping factor, $\mathbf{f}_{int}$ the internal forces on X, $\mathbf{f}_{ext}$ the external forces applied on X. γ must not be too low to escape oscillations and not too high to avoid slow displacements [1]. After numerous experiments, we have chosen to integrate equation (4) by Runge-Kutta's method. This method is slower than a simple Euler's method, but presents a better behavior when internal constraint are strong. The robustness of this method is nevertheless limited by the topological transformations which are unpredictable.

Unlike *snake*-like models, we do not perform any energy minimization. In fact, our approach shares several similarities with an energy minimization, because minimization is often performed with the associated Euler-Lagrange equation. It is easy to see that the two internal forces which we defined in Section 3.7 correspond to the regularization terms of first and second order of *snakes*. The "energy" of the mesh is thus minimal when it reaches a stable position. Several models use an approach similar to ours [9, 30].

3.6 Evolution

Depending on the application, the triangulated surface can be initialized with any number of non-intersecting icosahedra scattered in the volumetric image, or for instance, one icosahedron including the whole volumetric image. The surface may then optionally be refined (see section 3.4) until the density of the mesh corresponds to the application. After that, the surface is free to evolve according to its dynamic and geometrical rules.

The algorithm carrying out the evolution of the surface can be summarized as an iteration of the pseudo-code procedure of Figure 9. Note that vertex movements have to be bounded to check the geometric constraints correctly. Two methods can be implemented to limit them at each iteration: one can impose an upper bound to the speed of each vertex or the time scale (used to integrate the evolution equation) can be adjusted according to the speed of the fastest vertex. For the purpose of shape recovery, the first method can be preferred. For a correct physical behavior, it is better to choose the second one.

A simple heuristic is used to process motionless or slow vertices less frequently than mobile ones. It is indeed useless to check the geometric constraints of these vertices at every iteration. The processing periodicity of each vertex is thus made dependent on its speed. A maximal periodicity of 50 is imposed. The periodicity is one when the vertex movement is greater than 0.1δ. Between these upper and lower bounds, the periodicity is linear for the speed. This heuristic is particularly efficient in a multi-resolution approach, where lots of vertices are quickly near their final position. Note that the dynamic of every vertex is computed at each iteration, regardless of its speed.

3.7 Internal forces

We define two internal forces that depend on the neighborhood of each vertex: a force $\mathbf{f}_c$ of curvature regularization which smoothes the shape, and a spring force $\mathbf{f}_e$ which spreads localized deformations along the whole surface (it is the classical spring force when the rest length is null). If X (resp. Y) is a vertex, then $\mathbf{x}$ (resp. $\mathbf{y}$) expresses its coordinates, $\bar{\mathbf{x}}$ (resp. $\bar{\mathbf{y}}$) designates the mid-point of all the neighbors of X (resp. Y). We define at each vertex the following force:

$$\forall X \in S_{\mathcal{T}},\ \mathbf{f}_c(X) = \alpha_c \left(\bar{\mathbf{x}} - \mathbf{x} - \frac{1}{d(X)} \sum_{Y \in \mathcal{V}(X)} (\bar{\mathbf{y}} - \mathbf{y}) \right), \tag{5}$$

where α_c is the "rigidity" coefficient. Let $d_{\mathcal{T}}$ be the edge rest length for the whole mesh. This parameter can be set to null to minimize the area of the mesh; it can be set to a value given by the user to force the model to adapt the length of all edges; it can be set to the average of all edge lengths to regularize them along the entire mesh. The following force is defined at each vertex:

$$\forall X \in S_{\mathcal{T}},\ \mathbf{f}_e(X) = \alpha_e \sum_{Y \in \mathcal{V}(X)} (\|\mathbf{y} - \mathbf{x}\| - d_{\mathcal{T}}) \frac{\mathbf{y} - \mathbf{x}}{\|\mathbf{y} - \mathbf{x}\|}, \tag{6}$$

where α_e is the "stiffness" coefficient.

These two forces follow the action/reaction principle. The first one brings back vertices to their local tangent plane and minimizes surface curvature (it simulates thin plate behavior). The second one regularizes the edge lengths along the whole surface and expresses the binding energy. If the rest length is set to null, then the model tends to minimize its area (the model acts as a membrane). Note that a non-null rest length with strong elastic force can make the system rather unstable (it is also true in the 2D case, as was stressed for *snakes* by [26]).

3.8 External constraints

Shape recovery is one of our main purposes. To perform this task we introduce two external forces on vertices. They represent the influence of the image on the embedded surface. The force $\mathbf{f}_I$ will guide the surface towards an iso-potential value of the image. The force $\mathbf{f}_{\nabla I}$ will direct the surface to regions of maximal or minimal intensity value. For the applications we present here, external forces are not computed by a contour tracking or reconstruction algorithm such as in [32]. Moreover, the image is not pre-processed, and forces are not computed using a local scanning of the voxels surrounding the vertices [33]. External forces are just computed from the raw data. Both forces are normalized by the geometrical invariant δ (see section 3.2), so that the image influence is proportional to the mesh density.

The discrete volumetric image I is transformed into a continuous scalar field Π_I, called *image potential field*, by a tri-linear interpolation. This potential field is normalized to $[0, 1]$. The attraction towards an intensity value in this field is simply expressed by:

$$\forall X,\ \mathbf{f}_I(X) = \delta \alpha_I \left(\pi_I - \Pi_I(\mathbf{x}) \right) \mathbf{n}_X, \tag{7}$$

where α_I is the coefficient of attraction toward a given iso-potential surface of value π_I, and $\mathbf{n}_X$ is an approximation of the normal vector at vertex X. The force $\mathbf{f}_I$ is meant to search for the iso-potential surface of value π_I. Its principle is to inflate or deflate locally the model as long as it does not lie on the desired iso-potential surface. A positive value is expected for the coefficient α_I when the potential field tends toward one *ad infinitum* (objects are composed of

voxels whose intensity value is lower than α_I), a negative one when it tends toward zero (objects are composed of voxels whose intensity value is higher than α_I).

The discrete vector image ∇I is the discrete gradient of I computed by a Sobel operator. It is bounded by a maximal value given by the user, then transformed into a continuous vector field $\mathbf{\Pi}_{\nabla I}$ by tri-linear interpolation. The following force moves the surface along the local gradient of the image:

$$\forall X,\ \mathbf{f}_{\nabla I}(X) = \delta\left((\alpha_{\nabla I} - \beta_{\nabla I})(\mathbf{\Pi}_{\nabla I}(\mathbf{x}) \cdot \mathbf{n}_X)\mathbf{n}_X + \beta_{\nabla I}\mathbf{\Pi}_{\nabla I}(\mathbf{x})\right), \tag{8}$$

where $\alpha_{\nabla I}$ (resp. $\beta_{\nabla I}$) is the coefficient of gradient attraction along $\mathbf{n}_X$ (resp. $\mathbf{n}_X{}^{\perp}$). The force $\mathbf{f}_{\nabla I}$ is a classical gradient ascent when $\alpha_{\nabla I}$ and $\beta_{\nabla I}$ are equal and positive. The coefficient $\alpha_{\nabla I}$ modulates this force along the local surface normal, and $\beta_{\nabla I}$ along the local tangent plane.

This force can simulate the external energy of a *snake* model. Let J be the image of interest. Let I be the norm of the gradient image of J (possibly convolved with a Gaussian kernel). It is easy to see that our model is attracted to strong contours of image J when it is guided by force $\mathbf{f}_{\nabla I}$ on image I (with positive coefficients). This approach to edge finding could certainly be improved by using an edge image (computed by a Canny-Deriche operator for instance).

3.9 Example on a potential function

Figures 7a-e exhibit the behavior of the model during the shape extraction from a potential function: the expected shape is a chain with two intertwined rings. This potential function is similar to a distance transform image, which represents the distance to the torus skeletons. The model is initialized with a refined icosahedron including the whole shape. In these figures, the shape was extracted with external force $\mathbf{f}_I$ (parameters $\alpha_I = -1.0$ and $\pi_I = 0.5$) together with smoothing internal constraint $\mathbf{f}_c$ ($\alpha_c = 0.1$) and regularization constraint $\mathbf{f}_e$ ($\alpha_e = 0.1$). Ninety iterations are necessary for the surface to lie precisely on the shape. We also initialized the model with a set of $11 \times 11 \times 11$ small bubbles and we have run the process on the same potential function, but with no internal constraint. Figures 8a-f show that the model is robust compared with its initialization.

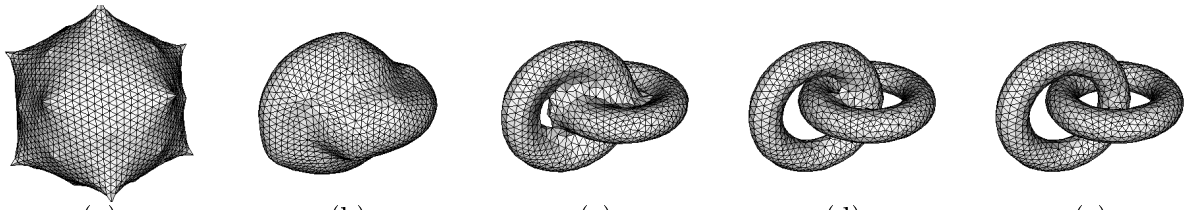
4 Image workspace and pyramids

4.1 Multi-scale approach with 3-D pyramids

A straight approach to image segmentation is not fully satisfactory. The influence of a potential function derived from an image is indeed localized around vertices (according to the definitions of the external forces $\mathbf{f}_I$ and $\mathbf{f}_{\nabla I}$) and does not make sense if the mesh has a resolution lower than the resolution of the three-dimensional image. The following two approaches can be taken:

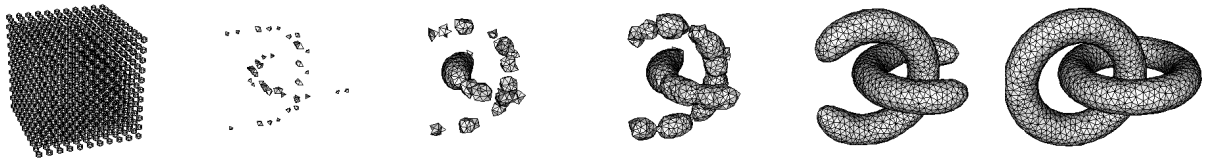
- The first one consists in using a triangulated mesh with a density comparable to the resolution of the image. The surface is then consistent with the frequency domain in which it evolves. One drawback is the need of using a very fine surface: the computational cost is increased accordingly.
- The second one does not make any assumption about the resolution of the mesh. Forces are computed from the image by a local scanning over a sufficiently large neighborhood of voxels. A pre-processing on the image can improve this approach [33].

In order to take advantage of both solutions, we propose to compute only once the influence of the image areas at different scales. This hybrid solution can be done by computing a three-dimensional image pyramid, where each resolution (i.e. each image) corresponds to distinct



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

Figure 7: Mesh evolution during the recovery of a chain shape composed of two intertwined tori: (a) at initialization; (b) at iteration 20; (c) at iteration 40; (d) at iteration 60; (e) at iteration 90.



(a) (b) (c) (d) (e) (f)

Figure 8: Mesh evolution during the recovery of a chain shape composed of two intertwined tori: (a) at initialization; (b) at iteration 10; (c) at iteration 30; (d) at iteration 40; (e) at iteration 70; (f) at iteration 160.

refinement of the triangulated mesh. The model will rely on the results obtained at a coarser resolution in order to start the computation at a finer resolution with more efficiency.

Pyramidal image representations as proposed in [37] have been the first ones to define and exploit image reduction. However several purposes may be sought, among which are the fast computation of parameters, compression, signal decomposition, segmentation, etc [19].

Pyramids of frequency decomposition presented in [3, 4] are more interesting for our purpose: they provide a set of images at decreasing resolutions which are closed to the visual perception of an observer at an increasing distance. The application of a Gaussian kernel filters high frequencies. After this filtering, a sampling of lower resolution provides an 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 extend pyramids of frequency decomposition to volumetric images composed of non-cubic voxels, and to link them together with our model of surface representation. Because the preceding pyramids are not always suited to 3D applications that are based on the embedding of a triangulated mesh into data, we have developed an algorithm for creating volumetric pyramids of any reduction factor. Therefore, the adequacy between the density of the embedded mesh and the resolution of the pyramidal image is preserved.

4.2 3D image pyramids of any reduction factor

The algorithm of pyramid construction we propose here does not assign a specific value to the reduction factor. Thus, any refinement can be used for the triangulated mesh. For instance, the refinement presented in section 3.4 requires a non-rational reduction factor of $\sqrt{3}$. We can notice that the authors of [35] have adapted the construction mechanism of discrete pyramids to allow rational reduction factors. However, the so-defined transformation is not a convolution process. Consequently, the filters are not low-pass ones and the resulting signals are not well defined. In

```
Procedure Evolution ( Mesh & T, const Image & I )
    for each Vertex U ∈ T,
        compute U.f_int(T) and U.f_ext(I)
    for each Vertex U ∈ T,
        application of the Newtonian law of motion on U with the previously computed Δt,
        U.f_int and U.f_ext; make effective movement of U.
    ListOfVertex L ⟵ all vertices of T
    Boolean x ⟵ false
    repeat
        while L.isNotEmpty() do
            Vertex U ⟵ L.popVertex ()
            for each Vertex V ∈ U.neighborhood ()
                for (U, V) check constraints (1) and (2); perform transformation (creation,
                deletion, inversion, annular transformation) accordingly; for each Vertex W
                involved in transformation, L.putAtEnd (W)
            end for
        done
        Update T.pointOctree and extract pairs ( Vertex U, Vertex V ) which do not satisfy
        constraint (3)
        if ∄(U, V) then x ⟵ true
        else
            for each (U, V), perform axial transformation; for each Vertex W involved
            in the transformation, L.putAtEnd(W)
        endif
    until x
end
```

Figure 9: This procedure describes the main steps of one iteration of deformation.

order that the consistency of the filtering/re-sampling operation be verified, the reduction factor per dimension. denoted ρ, must be less than 2.

We will first recall the construction of a classical Gaussian pyramid. The successive levels of that kind of pyramid are computed with the convolution of a Gaussian kernel of side 5 pixels (or voxels). It guarantees a low cost filtering without a phase translation linked to a reduction factor of two for each image dimension [6].

Let I_0 be the initial image of 3D voxels and the base of the pyramid. The computation of I_{h+1} (image of level $h+1$ in the pyramid) from I_h (image of level h in the pyramid) is given by the discrete convolution formula:

$$I_{h+1}(i',j',k') = \sum_{m=-2}^{2}\sum_{n=-2}^{2}\sum_{p=-2}^{2} \omega(m,n,p)\cdot I_h(2i'+m, 2j'+n, 2k'+p), \tag{9}$$

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, two major constraints have to be taken into account: voxels are not bound to be cubic (sampling frequencies are highly dependent on the acquisition devices and are not identical in the general case), the reduction factor of the re-sampling must be coherent with both the surface representation and its refinement.

Therefore the previous formulation (9) is not usable as is.

Making our voxel space isotropic in order to apply convolution operators coherently would be very memory intensive (the resolution would become the lowest common multiple of the sampling frequencies). Instead, by defining a real continuous workspace corresponding to the discrete structure containing the initial data, we will realize the convolution operations efficiently. In the following, a *discrete image* refers to the discrete volumetric data structure whose nodes (i.e. *voxels*) store intensity values. A *real image* designates the continuous scalar field obtained by the "embedding" of the discrete image into a subset of the Euclidean space: this "embedding" defines a *real size* for the image, which is generally not proportional to its *discrete size*. Intensity values in this field are computed by tri-linear interpolation. Images intensities are supposed to be normalized to $[0, 1]$.

Our goal is to determine a list of volumetric discrete images $I_0, I_1, \ldots, I_m$ representing the three-dimensional pyramid. I_0 is the initial image (i.e., the image I given for processing) of discrete size (M, N, P) and of real size (μ, ν, π). This image has the greatest amount of information. I_m 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 I_h for h between 0 and m. Their values are still unknown. Let E_h be the Cartesian space $M_h \times N_h \times P_h$. With these definitions, a discrete image I_h is a function from E_h towards $[0, 1] \subset \mathbb{R}$. Let $V_0, \ldots, V_m$ be the pyramid of real images corresponding to the pyramid of discrete images. Any real image V_h is given by the embedding then by the interpolation of the discrete data of I_h (i.e., $V_h = \Pi_{I_h}$). Every so-defined embedding preserves the real size (μ, ν, π) of the initial image I_0, because these images are meant to represent the same image at different scales.

We denote $\mathbb{E}$ the space defined by the real image of size $[0, \mu] \times [0, \nu] \times [0, \pi]$, which is a subset of $\mathbb{R}^3$. Because each I_h represents at different scale the same real image, they all have a real size of μ, ν, π. The embedding of a voxel (i, j, k) of a discrete image I_h into the real image space $\mathbb{E}$ is given by the transformation $\mathcal{T}_h$ (depending on the level of the pyramid) as below:

$$\begin{array}{rccl} \mathcal{T}_h : & E_h & \to & \mathbb{E} \\ & (i,j,k) & \mapsto & \left((i+\frac{1}{2})\frac{\mu}{M_h}, (j+\frac{1}{2})\frac{\nu}{N_h}, (k+\frac{1}{2})\frac{\pi}{P_h}\right) \end{array} \tag{10}$$

We call *unit* of the real space and we denote U_h the value $\min(\frac{\mu}{M_h}, \frac{\nu}{N_h}, \frac{\pi}{P_h})$. It is the smallest distance between the embedding of two voxels in the real image. In the case of an anisotropic image, the convolution mask applied during the construction must indeed be isotropic with respect to the real space where the image is embedded. If this is not properly done, pyramids will tend to preserve the contours following a direction where image resolution is fine, and to smooth too much those following a direction where image resolution is proportionally coarse. The unit U_h provides the isotropic distance separating the points of the convolution mask.

The discrete sizes M_h, N_h, P_h and the measure unit U_h correspond to a discrete image I_h and its associated real image V_h. Their values are defined recursively as below:

$$\begin{array}{cccc} M_0 = M & N_0 = N & P_0 = P & U_0 = \min(\mu/M, \nu/N, \pi/P) \\ M_{h+1} = \left\lfloor \frac{M_h}{\rho} \right\rfloor & N_{h+1} = \left\lfloor \frac{N_h}{\rho} \right\rfloor & P_{h+1} = \left\lfloor \frac{P_h}{\rho} \right\rfloor & U_{h+1} = \rho U_h \end{array} \tag{11}$$

Let $R = (i', j', k')$ be a voxel of the discrete data of I_{h+1}. Our goal is to find its value for any $(i', j', k') \in E_{h+1}$. Its embedding $R_{\mathbb{E}}$ in the real image V_{h+1} has coordinates of $\mathcal{T}_{h+1}(i', j', k')$ (see Figure 10a).

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 3D for a kernel of size 5) is determined with the unit U_h: V_h is thus discretized around the point $R_{\mathbb{E}}$ (see Figure 10b). Supposing I_h is known, then V_h is

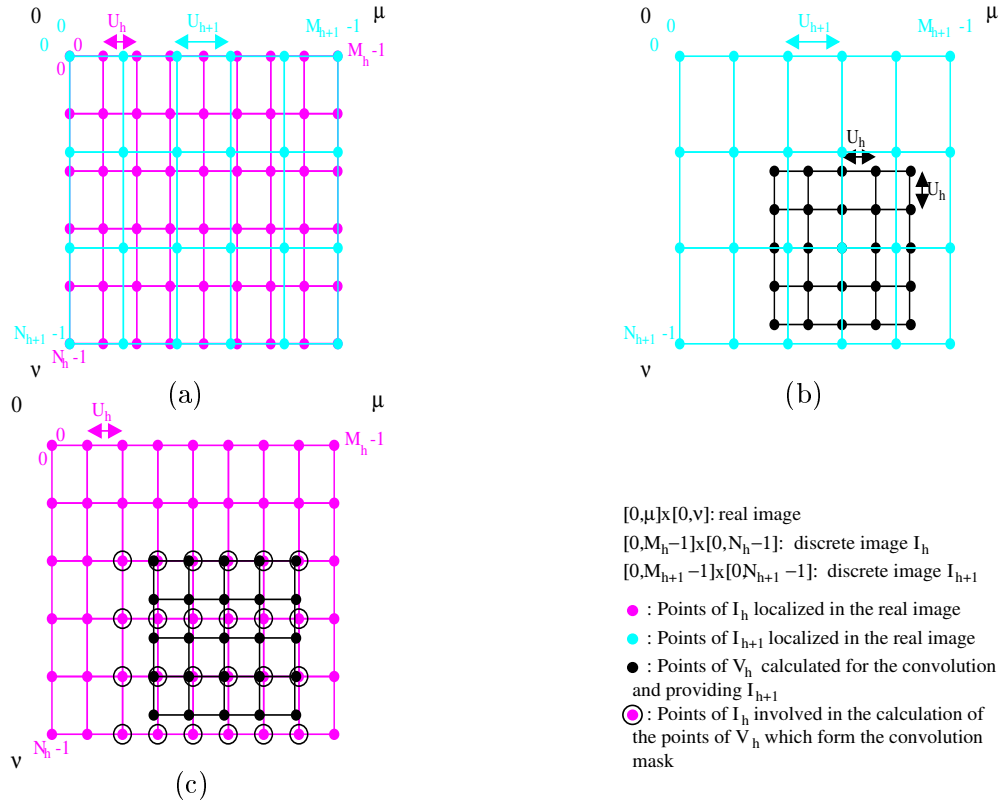


Figure 10: View of a convolution computation in 2D: (a) the two superposed levels I_h and I_{h+1}; (b) computation of level I_{h+1} together with the localization of the convolution mask applied to one point; (c) application of the convolution mask over level I_h, discrete points of I_h involved in the computation are also displayed with black circles.

defined by Π_{I_h}. We obtain the convolution formula:

$$I_{h+1}(i',j',k') = \sum_{m=-2}^{2}\sum_{n=-2}^{2}\sum_{p=-2}^{2} \omega(m,n,p) V_h\left(\mathcal{T}_{h+1}(i',j',k') + (mU_h, nU_h, pU_h)\right). \qquad (12)$$

To compute points at the boundary of I_{h+1}, the boundary voxels of image I_h are replicated.

Because of the unknown reduction factor, the 5^3 points involved in the convolution do not coincide with given points of I_h in the general case (see Figure 10c). Each one of these points is computed with a tri-linear interpolation from the 8 data points of I_h which surround it.

The Gaussian convolution kernel (of size 5^3) is applied successively along the three dimensions because of its separable property. We can estimate the savings offered by this optimization (the following notations are used: let t' be the access time to a point value, t_1 the running time of a classical algorithm, t_2 the running time of the optimized algorithm):

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

$$\text{Hence, } \frac{t_2}{t_1} = \frac{1}{25}(\rho^2 + \rho + 1). \qquad (13)$$

The optimized algorithm is thus faster when the reduction factor ρ is between 0 and $\frac{-1+\sqrt{97}}{2}$ (about 4.4). An overview of the 3D pyramid construction algorithm is given in appendix A.

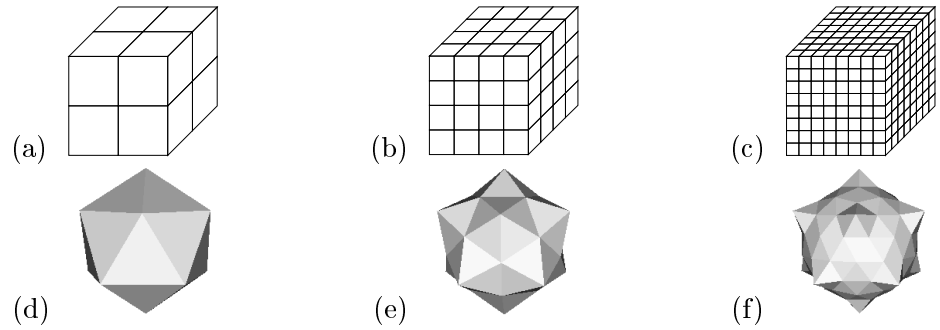


Figure 11: Resolution of a pyramidal image and density of a triangulated mesh: (d) evolves in (a), (e) in (b), (f) in (c).

4.3 Image-model relation

In section 4.1 we have chosen to express surface-image interaction with forces locally computed around each vertex. Using a pyramid requires a model sufficiently flexible to adapt its density to image resolution. The edges of our model should neither be too long, otherwise high frequency contours could be missed, nor too small, because they would then represent the decomposition of a small contour of two voxels. In order to obtain a correct adequacy between the surface and the images of the pyramid, we first examine the relations linking the model density to the resolution of an image, then we show how to maintain the surface–image adequacy during the whole coarse-to-fine process.

According to constraints (1) and (2), the mesh density is defined by the invariant δ. The coarse-to-fine approach implies a refinement of the model every time it goes down a level of the pyramid (see Figure 11). The invariant δ is thus dependent on the image of the pyramid in which the model is currently evolving (i.e, δ is a function of the level h). Let δ_h be the invariant δ of the mesh at level h of the pyramid. Let d_h (resp. D_h) be the minimal (resp. maximal) edge length of the mesh at level h. Constraints (1) and (2) give $d_h = \delta_h$ and $D_h = 2.5\ \delta_h$.

The image resolution is closely linked to the unit U_h. In this section, we suppose that images are isotropic (see Section 4.4 for anisotropic images). Edges of the mesh represent discrete contours of the voxel image. Both 6-connected contours and 26-connected contours are likely to have corresponding edges. Consequently, an edge of the mesh may be smaller than two 6-connected voxels, which implies $d_h \leq U_h$, and may be longer than two 26-connected voxels, giving $D_h \geq \sqrt{3}\, U_h$. Hence,

$$\frac{2.5}{\sqrt{3}} \geq \frac{U_h}{\delta_h} \geq 1. \tag{14}$$

Equation (14) constraints mesh density as a function of image resolution.

A surface of given invariant δ may be built only at initialization. After that, modifications of the invariant can be limited by the current mesh geometry. The refinement transformation $\Delta_{\frac{1}{\sqrt{3}}}$ (see Section 3.4) reduces the average edge length to $1/\sqrt{3}$ of the old one. Therefore we apply to the invariant a reduction factor whose value is $\sqrt{3}$. In order that the inequality (14) be respected at the initialization and during all successive levels, an identical reduction factor is chosen for the pyramid construction; thus δ_h and U_h are recursively defined by:

$$\rho = \sqrt{3} \text{ and } \forall h = 0 \ldots m-1, \quad \left\{ \begin{array}{c} \delta_m = \delta_{init},\ \delta_h = \delta_{h+1}/\rho \\ U_0 = U,\ U_{h+1} = \rho U_h \end{array} \right. \tag{15}$$

At the initialization moment, a bubble or a set of bubbles, whose invariant δ_{init} is consistent with (14) at level m, is created. During the evolution and the coarse-to-fine process, definitions

of (15) ensure a correct surface-image adequacy whichever are the iteration or the current level in the pyramid (i.e., δ_h and U_h follow (14)). The time when the mesh is refined and the resolution increased is determined by a criterion based on a motion estimate of the mesh (refer to Section 5.1).

4.4 Mesh evolution in an anisotropic image

During the segmentation process, edges of the mesh have to keep their meaning with regards to the voxel space. On one hand, if we decide to work in the real image with size (μ, ν, π), edges lose their consistency with respect to the resolution of data. On the other hand, mesh evolution in a real image where voxels are cubic modifies the constraints that must be applied: forces lose their physical interpretation. Three different ways can be outlined to tackle this problem:

- The surface evolves in a real space of size (μ, ν, π) and follows the physically-based constraints. Surface-image consistency is achieved only on the axes of fine resolution.
- The surface evolves in a real space defined from the space (μ, ν, π) by affine transformation. This space has the same proportions than the discrete image it interpolates (its sizes is $(\tau M_h, \tau N_h, \tau P_h)$). The behavior of internal forces is slightly different from the corresponding forces in the real physical space.
- The surface evolves in a real space of sizes (μ, ν, π). An anisotropic metric is coupled with this space. This metric is defined from the current discrete image (M_h, N_h, P_h): adequacy between the surface and the image is achieved along all axes and internal forces keep a physical meaning.

The first method gives good results with a weak anisotropy; the second one provides better results when the anisotropy is more significant; the last one is theoretically the best solution whichever is the context but has the slowest implementation. For most applications dedicated to volumetric data analysis, exact physical behavior is not critical and the second method is preferred to the latter.

5 Implementation and results

5.1 Algorithms of surface extraction

Figure 12 presents the algorithm of shape recovery on an image and Figure 13 presents the coarse-to-fine algorithm on a pyramid of images. The convergence criterion, which decided when the mesh is refined and goes down one level in the pyramid, is the average kinetic energy along the normal to the surface (the other part represents the sliding of vertices over the surface). This energy is normalized by the invariant δ_h and also by the time step. This criterion may optionally be sharpened with a maximal speed check or a validation by user interaction. Other criteria may be added easily.

The model is tested on a synthetic fractal image (the classical Sierpinski's cube) to point out both topological transformations and multi-resolution approach. The image size is $81 \times 81 \times 81$. The topology of the shape to recover is highly complex and unpredictable. The model has the expected behavior which is to extract first areas of higher density (see Figure 14). The physical parameters were set to the following values: $\alpha_c = 0.05$ and $\alpha_e = 0.001$ for internal forces, $\alpha_I = -1.0$, $\pi_I = 0.4$, $\alpha_{\nabla I} = 0.0$ and $\beta_{\nabla I} = 0.0$ for external forces. Note that π_I is slightly decreased for the uppermost levels of the pyramid: the fractal object has indeed an empty volume *ad infinitum* and therefore a null density.

```
Procedure RecoverShape ( Mesh & T, const Image & I, const double ε )
    /* Adequacy surface-image */
    while T.δ > I.U do
        T.globalRefinementΔ_{1/√3}()
    done
    /* Deformation until a stable position is achieved */
    repeat
        Evolution (T, I)
        double E ←— T.computeKineticEnergyAlongNormals()
    until E < ε
end
```

Figure 12: Algorithm of shape recovery over a given image. The mesh T given as initialization is refined as long as its density is not consistent with the resolution of image I.

```
Procedure PyramidalRecoverShape ( Mesh & T, const PyramidOfImage & P,
                                  const double ε, const int m )
    int i ←— m
    while i >= 0 do
        RecoverShape( T, P.image(i), ε )
        i ←— i − 1
    done
end
```

Figure 13: Shape recovery with a pyramidal approach: m is a given level in the pyramid P of images. The mesh evolves in each image P.image(i) of the pyramid with i from m to 0. The mesh T is given as an initialization on the coarser level of the pyramid. After convergence on level i, the result (i.e., T) is given as initialization for level $i-1$. Refinement is done in procedure *RecoverShape*().

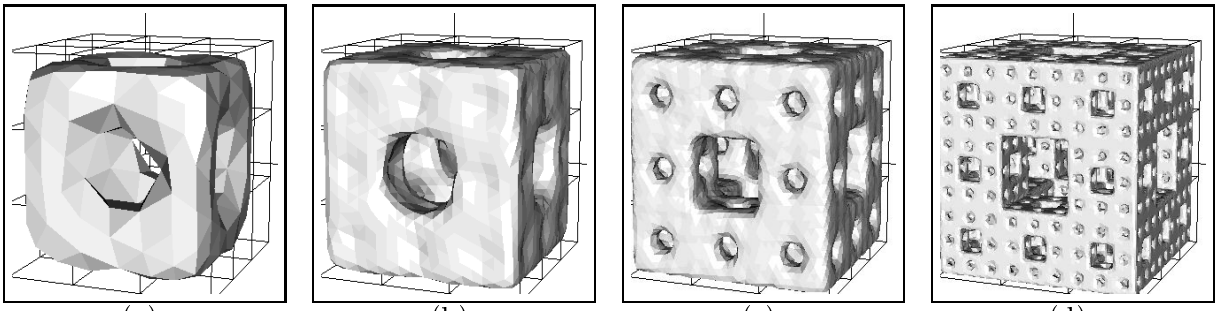


Figure 14: Multi-resolution evolution of the mesh on a synthetic image (fractal volume of Sierpinski): (a) after convergence on image I_3; (b) after convergence on image I_2; (c) after convergence on image I_1; (d) final result on I_0.

5.2 Comparison of the two approaches on medical data

We compare the two approaches on a computed tomography of a head. Figure 19a displays a volumetric rendering of this data (each intensity value has an associated opacity). The discrete size of the image is $256 \times 256 \times 68$ and its real size is $1.0 \times 1.0 \times 1.0625$. The image is significantly anisotropic, thus the second method described in section 4.4 is used to process it. Physical parameters are set to the following values: $\alpha_c = 0.4$ for a gentle smoothing of the surface, $\alpha_e = 0.0$ because regularizing edges is not a critical point in our application, $\alpha_I = -1.0$ and $\pi_I = 0.1$ to track bone intensity value, $\alpha_{\nabla I} = 0.0$ and $\beta_{\nabla I} = 0.0$ because gradient maxima do not correspond to the shape to recover.

A pyramid P composed of images $I_0, \ldots, I_m$ $(m = 3)$ is built up from this data set with a reduction factor of $\sqrt{3}$. The process is run twice for comparison purposes on a *Pentium* $300Mhz$ *with* $128Mb$ *of memory*:

- The process is run first on the volumetric image I_0 without multi-resolution by calling the procedure *RecoverShape*() with a bubble including the whole image. Figure 15 shows the surface evolution: at first, the surface is automatically refined (at this time, the mesh has more than 65000 vertices), then the surface slowly sticks on the outer part of the skull, and afterwards goes inside to recover its inner part (orbits of the eyes, brain cavity, etc). More than 700 iterations are necessary for the surface to rest perfectly on the inner part of the skull.

- The process is now run on the pyramid P at level m: procedure *PyramidalRecoverShape*() is called with the same mesh at initialization. The process waits for the convergence at one pyramid level before going down one level. Figure 16 displays surface evolution with a multi-resolution method. The surface, at first coarse, quickly outlines the skull shape. Then, it relies on the shape extracted at one level to start the evolution on next level as near as possible of the expected result.

The Figure 17 compares the behavior of both methods. The behavior of the first one (direct approach) is clear. The kinetic energy curve shows the slow convergence of the model (see Figure 17a) and the small variations of its number of vertices (see Figure 17b). At the beginning, the mesh has more than 65000 vertices and, at the end of the process, about 120000 vertices. The behavior of the second method (multi-resolution approach) is also displayed on theses figures, and points out the evolution in four levels of the pyramid: the mesh has goes down a level at iterations 400, 600, and 800. At the beginning, the mesh has only 6000 vertices, and more than 120000 vertices at the end. For this image, the surfaces obtained by these two methods have an area that differs by less than 1.0% and a volume that differs by 0.3%. These (very) small differences can be explained by the fact that the two surfaces may have stabilized in different local minima. Both triangulated surfaces have one connected component and twenty-three topological holes.

The Figure 18 shows the computation time of both methods and clarified the amounts of time spent by the computation of topological operations (detection and resolution), the computation of surface normals, and the computation of the model dynamics. Note that the computation time of topological operations and of normals slightly decreased along with the model convergence because of the heuristic presented in Section 3.6. The Table 1 displays the total computation time (in seconds) for the two methods.

The skull is outlined with 3800 vertices in less than one minute. Two minutes later, the skull shape is refined and has now 12000 vertices. Six minutes thirty seconds later, we have a skull model composed of more than 38000 vertices. Thirty more minutes are necessary to achieve convergence on the finest level (the model has more than 120000 vertices). The direct approach is nearly three times as long as the multi-resolution approach.

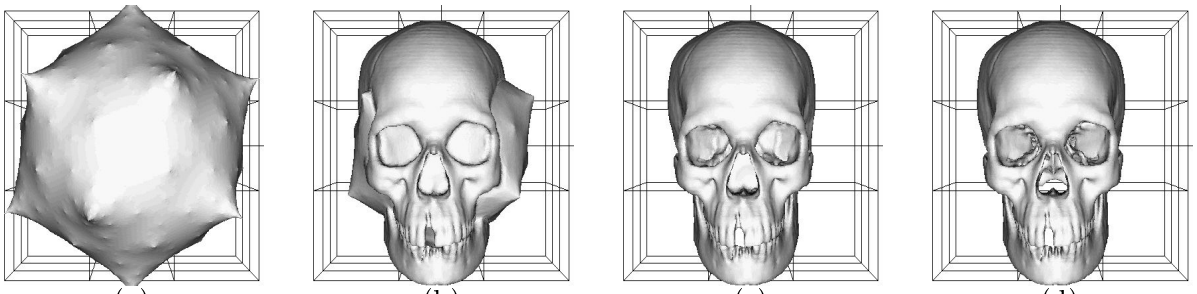


Figure 15: Surface evolution without a pyramidal approach (no Gouraud shading is done): (a) iteration 0 on image I_0; (b) iteration 200 on image I_0; (c) iteration 400 on image I_0; (d) iteration 1100 on image I_0.

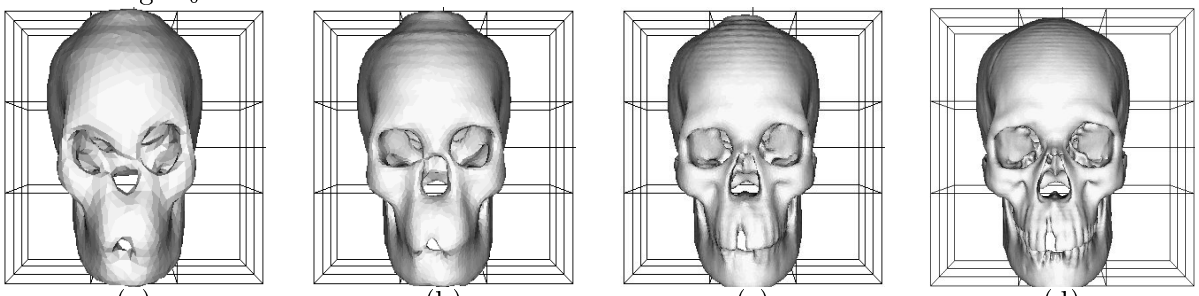
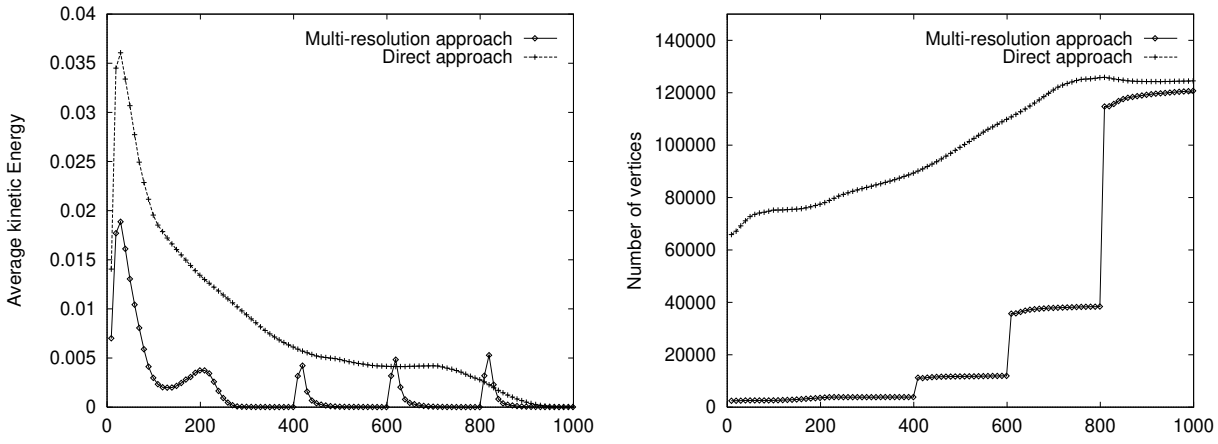


Figure 16: Surface evolution on a pyramid of images (no Gouraud shading is done): (a) iteration 399 on image I_3; (b) iteration 599 on image I_2; (c) iteration 799 on image I_1; (d) iteration 999 on image I_0.



(a) (b)

Figure 17: Two methods of shape extraction from volumetric data are compared on these graphs (with and without multi-resolution): (a) evolution of the average kinetic energy accumulated along the surface normals; (b) evolution of the number of vertices.

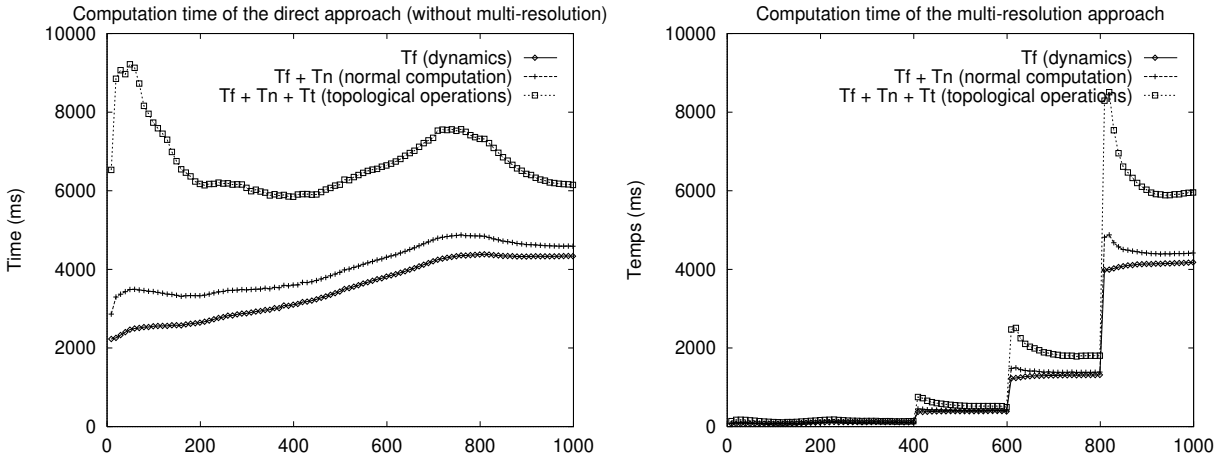
5.3 Other results

We have tested the robustness of our model and of the multi-resolution approach over different kinds of volumetric data. The second data set is a phase contrast MR angiographic image of the brain vessels and is more problematic for a pyramidal approach. The Figure 19b displays a volumetric rendering of this data. Its discrete size is $256 \times 256 \times 124$. Angiographic images

Table 1: Comparison between the direct approach and the multi-resolution approach: computation time on a computed tomography.

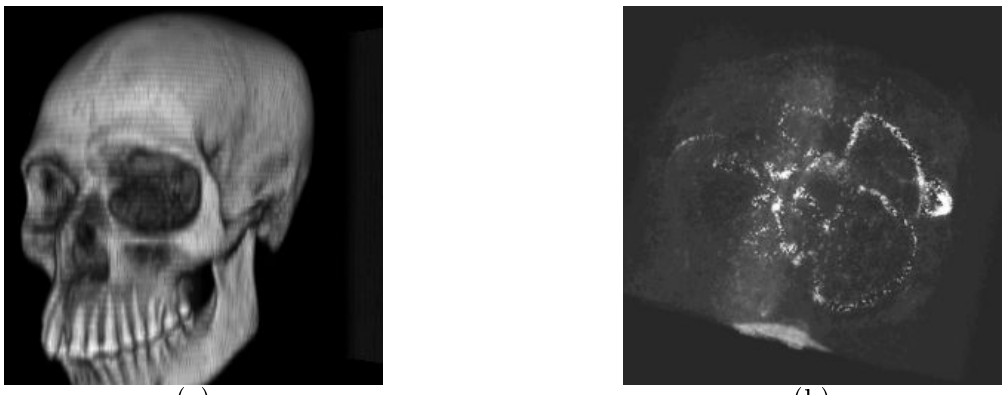
Image	Approach		T(s) on I_3	T(s) on I_2	T(s) on I_1	T(s) on I_0	Total
CT	multi-	T	17,0	30,0	108,7	543,8	$11min\ 39s$
	resolution	N	1,4	5,5	22,3	97,6	$2min\ 27s$
		F	38,8	77,8	258,3	1242,6	$26min\ 58s$
		=	$0min\ 57s$	$1min\ 53s$	$6min\ 30s$	$31min\ 24s$	$40min\ 45s$
CT	direct	T				2701,7	$45min\ 01s$
		N				553,8	$9min\ 13s$
		F				3476,4	$57min\ 56s$
		=				$112min\ 12s$	$112min\ 12s$

Symbols T, N, F respectively designate the computation time of topological operations, of normals, and of dynamics.



(a) (b)

Figure 18: Evolution of the computation time as a function of the iteration number: (a) computation time for a direct approach (in ms); (b) computation time for a multi-resolution approach (in ms). As it can be seen on both graphs, computing the movement of vertices takes the longest time and varies as a linear function of the number of vertices.



(a) (b)

Figure 19: Volumetric rendering of two medical data set: (a) a computed tomography of a skull; (b) a phase contrast MR angiography of the vessels of the brain.

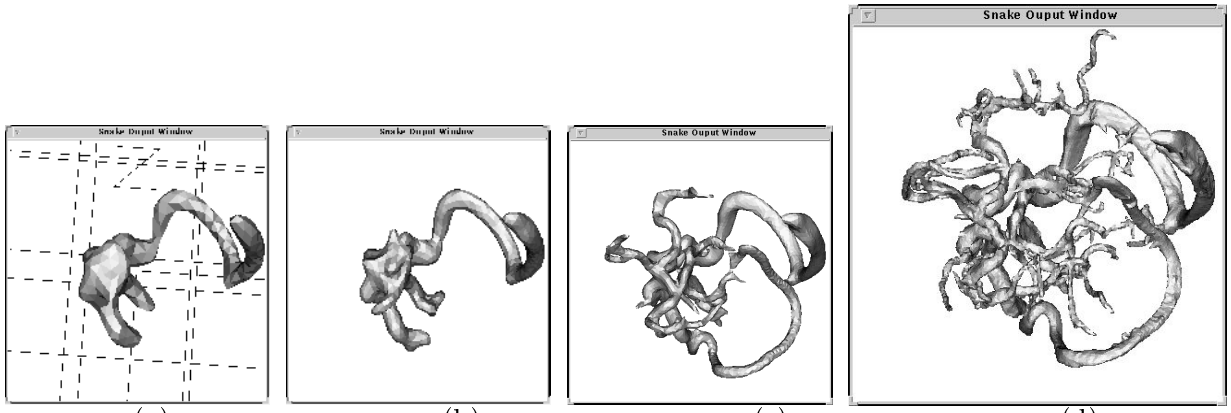
Table 2: Comparison between the direct approach and the multi-resolution approach: computation time on a MR angiography.

Image	Approach		T(s) on I_3	T(s) on I_2	T(s) on I_1	T(s) on I_0	Total
aRM	multi-	T	6,8	0,0	3,8	196,9	$3min\ 27s$
	resolution	N	1,4	0,0	0,0	26,4	$0min\ 28s$
		F	4,3	0,1	27,8	729,2	$12min\ 41s$
		=	$0min\ 13s$	$0min\ 0s$	$0min\ 32s$	$15min\ 52s$	$16min\ 37s$
aRM	direct	T				1521,0	$25min\ 21s$
		N				344,6	$5min\ 45s$
		F				986,4	$16min\ 26s$
		=				$47min\ 32s$	$47min\ 32s$

Symbols T, N, F respectively designate the computation time of topological operations, of normals, and of dynamics.

are highly contrasted. They are not suited to a pyramidal representation because vessels are thin objects: therefore they are composed of high frequency information and little information remains on the coarsest level of the pyramid. Consequently, the mesh extracts few data from coarse levels. As shown in Figure 20, the model succeeds in following the vessels: connected vessels are recovered on fine levels. Physical parameters were set to the following values: $\alpha_c = 0.07$ and $\alpha_e = 0.0$ for internal forces, $\alpha_I = -1.0$, $\pi_I = 0.05$, $\alpha_{\nabla I} = 0.0$ and $\beta_{\nabla I} = 0.0$ for external forces. The Table 2 displays the computation time on this database for the two approaches.

However, the surfaces obtained by these two approaches do not possess the same number of connected components. In fact, the multi-resolution approach has kept only one component (the only one with low frequency information) and has avoided several small components. The volume contained in these surfaces differs by 3.7%.

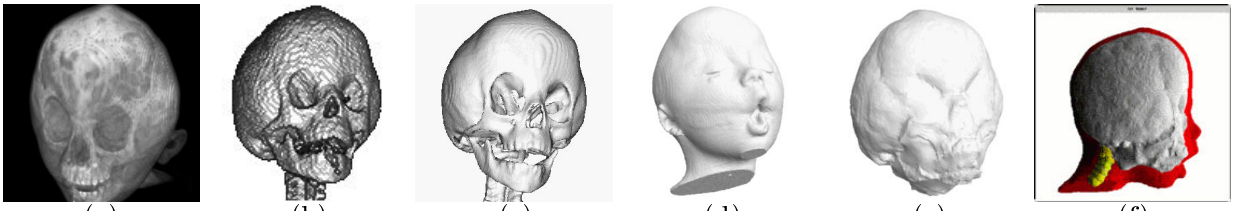


(a) (b) (c) (d)

Figure 20: Surface evolution during the recovery of brain vessels from an angiographic image with a pyramidal approach: (a) after convergence on image I_3; (b) after convergence on image I_2; (c) after convergence on image I_1; (d) final result on image I_0.

The model can also extract several structures from the same image. Figure 21a is a volumetric rendering of a computed tomography of a child head. Figure 21b is a surface rendering of the data with the iso-value 0.29. Figure 21c is the shape extracted by our model with force $\mathbf{f}_I$ and the same iso-value. Figure 21d displays the surface obtained with force $\mathbf{f}_I$ and parameters $\alpha_I = -1$ and $\pi_I = 0.1$. Figure 21e displays the surface obtained with force $\mathbf{f}_{\nabla I}$ and parameters $\alpha_{\nabla I} = 0.1$ and $\beta_{\nabla I} = 0.05$: the model has searched for maxima of intensity value and has thus rested on

the skull "surface" while filling the gaps in it (orbits, space between jaws, etc). Figure 21f is a mixed view of these shapes.



(a) (b) (c) (d) (e) (f)

Figure 21: Comparison between the two forces $\mathbf{f}_I$ and $\mathbf{f}_{\nabla I}$ on the same database (initialization is the same in all cases): (a) volumetric rendering; (b) surface rendering with iso-value of 0.29; (c) extraction of skull shape with $\mathbf{f}_I$ ($\pi_I = 0.29$); (d) extraction of skin contour using $\mathbf{f}_I$ ($\pi_I = 0.1$); (e) extraction of skull "hull" using $\mathbf{f}_{\nabla I}$; (f) mixed view of three shapes from the database: the contour skin, the skull "hull" and vertebrae.

In cellular imaging, it is sometimes necessary to mark only the boundary of the structures of interest. The recovery of components of that kind of images cannot be done with iso-surfaces. On the other hand, the force $\mathbf{f}_{\nabla I}$ can be used to extract shapes from such images because it seeks intensity maxima. We have tested our model on an image obtained by confocal microscopy, representing a nucleus of a polynuclear cell, and whose boundary has been marked with fluorescence.

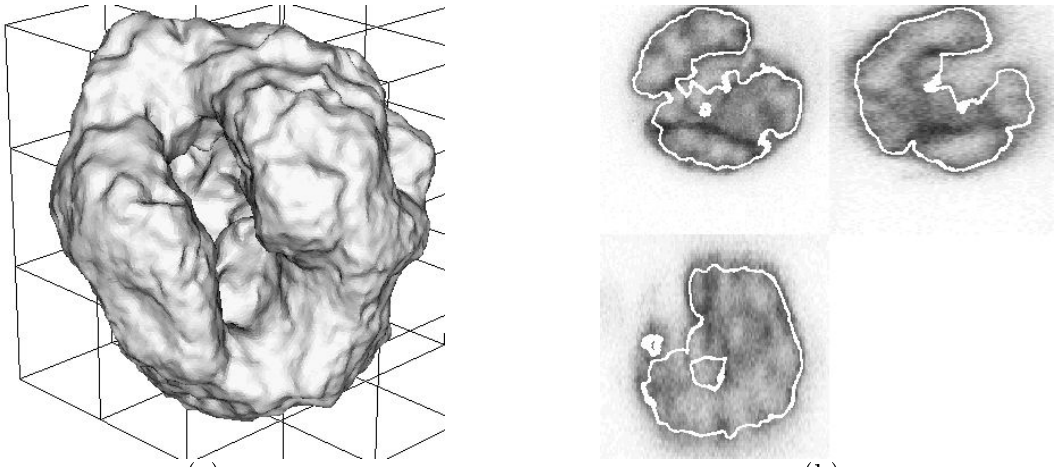
For this image, force $\mathbf{f}_{\nabla I}$ is used with coefficients $\alpha_{\nabla I} = 0.1$ and $\beta_{\nabla I} = 0.0$. A bubble is initialized around the image. In order that the bubble retract on the shape, the model undergoes a slight elastic force $\mathbf{f}_e$ ($\alpha_e = 0.2$) with a null rest length. The image is rather noisy, so we impose a regularization force $\mathbf{f}_c$ with coefficient $\alpha_c = 1.5$. The Figure 22a shows the surface obtained after convergence, and the Figure 22b shows the adequacy of the model to the data on three orthogonal slices.

The model can deform any closed and oriented triangulated surface. The result of a Marching-Cubes algorithm [27] can therefore be used as an initialization for our process. Figure 23a shows an extracted iso-surface from a computed tomography using an extended version of the Marching-Cubes [24], which ensures the closure and the orientability of the generated iso-surface. Figure 23b displays this surface after several iterations of our model parameterized with a "rigidity" constraint $\alpha_c = 0.3$. The iso-surface computed by the Marching-Cubes has 354 connected components and 958 topological holes (and about 295,000 vertices) whereas the deformed surface has only 45 connected components and 181 holes (and about 191,000 vertices). The introduction of inner forces has removed the most physically unstable parts of the surface. We stress that this is not a mesh simplification: the smoothing is physical, neither geometrical nor topological. Classical simplification algorithms can be used efficiently as a post-processing for our model but they are not suited to remove small artifacts; most simplification algorithms indeed tend to keep these artifacts and simplify quasi-planar regions [18] [42].

Iso-surface tracking can also be achieved efficiently: Figure 24a displays an image of a lymphocyte obtained by confocal microscopy. Figures 24b-d show the extracted shapes with an increasing parameter π_I.

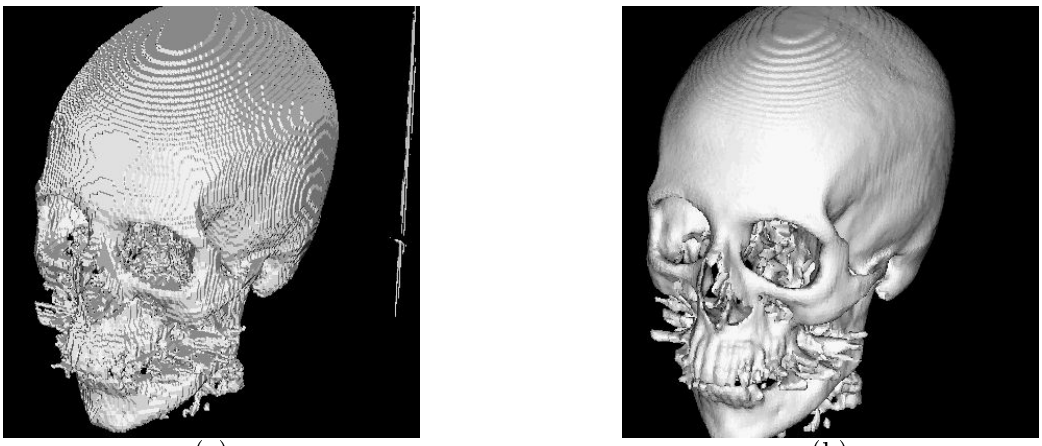
6 Conclusion

We have designed and developed an efficient model for shape recovery on volumetric images. This deformable model can automatically adapt its topology to the variation of its geometry for an acceptable computational burden: it takes about 5 seconds to detect and solve topo-



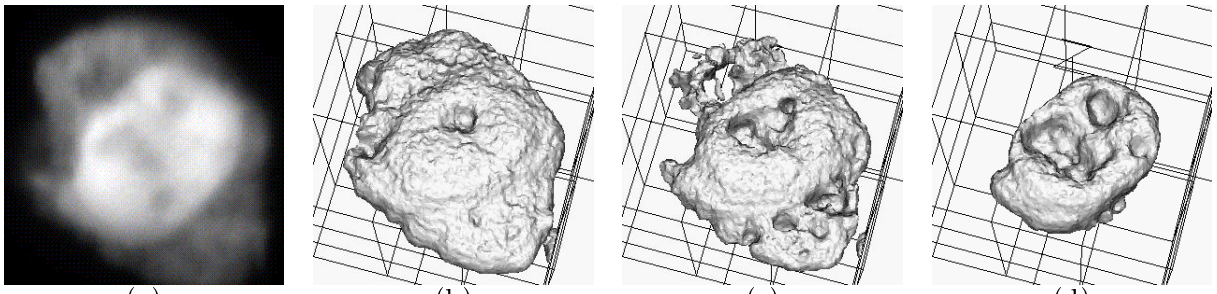
(a) (b)

Figure 22: Shape extracted from a confocal microscopy image with force $\mathbf{f}_{\nabla I}$: (a) view of the triangulated surface obtained after convergence (the mesh has about 41000 vertices, 1 connected component and 2 holes); (b) embedding of the surface into the image on three orthogonal slices (surface points are in white).



(a) (b)

Figure 23: Chaining of a Marching-Cubes with our model: (a) result of the Marching-Cubes algorithm; (b) after deformation under smoothness forces.



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

Figure 24: Iso-surface extraction on a lymphocyte image with variation of parameter π_I: (a) volume rendering; (b) parameter π_I is set to 0.3; (c) parameter π_I is set to 0.45, (d) parameter π_I is set to 0.75.

logical transformations over a mesh with 65000 vertices (and without the use of the heuristic), and less than 2 seconds with 125000 vertices when the heuristic optimizes 85% of the vertices (which is often the case with the multi-resolution approach). This model can be coupled with a three-dimensional image pyramid to quickly outline objects within the image. We have tested the model and the multi-resolution approach on various databases. Shape recovery results are encouraging and the coarse-to-fine process is extremely efficient on images that have relevant low frequency information. After the process is complete, any classical mesh reduction algorithm can be run to obtain a compact surface representation of objects.

Several points may nevertheless be explored:

- The convergence may be guided by some new constraints, for instance by introducing attractive forces generated either by attachment points of the object or by particular edges detected during a pre-processing. At the moment, we are working on wavelet pyramids in order to detect high contrasts and singularities of images. This pre-processing would speed up the convergence of the algorithm and provide a better detection of unstable parts of the surface.

- In order to widen the application scope of our model, the physical formulation can be complemented by adding global dynamic parameters, such as global speed or instant rotation vector [38].

- The sampling of the mesh is more or less uniform and is not optimized for quasi-planar regions. The simplex mesh model of [9] allows a non-uniform sampling according to the local curvature but topological breaks are neither detected nor performed any more. The challenge is to allow a non-uniform sampling to process fewer vertices without losing the speed of topological break detection given by a uniform sampling. Non-Euclidean metric system (i.e., distance computation may vary according to the location) is currently studied to combine both advantages.

Acknowledgements

The authors thank Dr. Yves Usson (TIMC-IMAG Lab., Grenoble, France) for providing the CT image of the skull and the confocal microscopy images. Our acknowledgements to the UMDS Image Processing Group, London, United Kingdom, for the phase contrast MR angiographic image. Acknowledgements for the CT image of the child to S. Lobregt, CT Scanner Science Department, Philips Medical Systems, The Netherlands, and Dr. F.W. Zonneveld, Department of Diagnostic Radiology, Utrecht University Hospital, Utrecht, The Netherlands; used with courtesy of Prof. J.C. van der Meulen, Department of Plastic and Reconstructive Surgery, Rotterdam University Hospital "Dijkzicht", Rotterdam, The Netherlands.

Many thanks to Jean-Marc Nicod and Serge Miguet (LIP, ENS Lyon, France) for providing a version of the *Marching-Cubes* algorithm. The authors are also grateful to Guillaume Berche, Marie-Laurence Hollett, and Nadège Saint Martin Tillet, for their careful reading of the paper. We thank the anonymous reviewers for their helpful and constructive comments.

References

[1] M.-E. Algorri and F. Schmitt. Surface Reconstruction from Unstructured 3D Data. *Computer Graphics forum*, 15(1):47–60, 1996.

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

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

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

[5] X. Chen and F. Schmitt. Intrinsic Surface Properties from Surface Triangulation. In G. Sandini, editor, *Proc. of European Conference on Computer Vision*, volume 588 of *Lecture Notes in Computer Science*, Santa Margherita Ligure, Italy, May 1992. Springer-Verlag.

[6] A. Chhikian. Algorithmes optimaux pour la gnration de pyramides passes-bas et laplaciennes. *Traitement du Signal*, 9:297–308, January 1992.

[7] L.D. Cohen and I. Cohen. Finite-element methods for active contour models and balloons for 2-D and 3-D images. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 15(11):1131–1147, November 1993.

[8] H. Delingette. Simplex Meshes: a General Representation for 3D Shape Reconstruction. In *Proc. of Int. Conf. on Computer Vision and Pattern Recognition*, Seattle WA, USA, 1994.

[9] H. Delingette. General Object Reconstruction based on Simplex Meshes. Research Report 3111, INRIA, Sophia Antipolis, France, February 1997.

[10] M. Eck and H. Hoppe. Automatic Reconstruction of B-Spline Surfaces of Arbitrary Topological Type. *Computer Graphics, Annual Conference Series (SIGGRAPH'96 Proceedings)*, pages 325–334, August 1996.

[11] N. Flasque, M. Desvignes, and M. Revenu. Coopération de modèles déformables pour l'imagerie cérébrale en 3 dimensions. In *Proc. of Seizième Colloque sur le Traitement du signal et des Images GRETSI*, pages 733–736, Grenoble, France, September 1997.

[12] J. Françon. Discrete Combinatorial Surfaces. *CVGIP: Graphical Models and Image Processing*, 57(1):20–26, January 1995.

[13] P.J. Frey and H. Borouchaki. Texel: triangulation de surfaces implicites. partie I: aspects théoriques. Research Report 3066, INRIA, Rocquencourt, France, December 1996.

[14] D. Gordon and J.K. Udupa. Fast surface tracking in three-dimensional binary images. *Computer Vision, Graphics, and Image Processing*, 45(2):196–241, February 1989.

[15] S. Gottschalk, M.C. Lin, and D. Manocha. OBBTree: A Hierarchical Structure for Rapid Interference Detection. *Computer Graphics, Annual Conference Series (SIGGRAPH'96 Proceedings, New Orleans, Louisiana)*, pages 171–180, 1996.

[16] H.B. Griffiths. *Surfaces*. Cambridge University Press, 1976.

[17] A. Guéziec. Surface representation with deformable splines: Using decoupled variables. *IEEE Computational Science and Engineering*, 2(1):69–80, mar 1995.

[18] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle. Mesh Optimization. *Computer Graphics, Annual Conference Series (SIGGRAPH'93 Proceedings)*, pages 19–26, August 1993.

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

[20] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active contour models. *International Journal of Computer Vision*, 1(4):321–331, 1987.

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

[22] J.-O. Lachaud and E. Bainville. A discrete adaptive model following topological modifications of volumes. In *Proc. 4th Discrete Geometry for Computer Imagery (DGCI'94), Grenoble, France*, pages 183–194, September 1994.

[23] J.-O. Lachaud and A. Montanvert. Volumetric Segmentation using Hierarchical Representation and Triangulated Surface. Research Report 95-37, Laboratoire de l'Informatique du Parallélisme, ENS Lyon, France, November 1995.

[24] J.-O. Lachaud and A. Montanvert. Segmentation tridimensionnelle hiérarchique par triangulation de surface. In *Proc. 10th Reconnaissance de Formes et Intelligence Artificielle (RFIA'96), Rennes, France*, pages 13–22, January 1996.

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

[26] F. Leymarie and M.D. Levine. Tracking deformable objects in the plane using an active contour model. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 15(6):617–634, June 1993.

[27] W. E. Lorensen and H. E. Cline. Marching Cubes: A High Resolution 3D Surface Construction Algorithm. *Computer Graphics*, 21(4):163–169, July 1987.

[28] D. MacDonald, K. Worsley, D. Avis, and A.C. Evans. 3D Mapping of Variability in Cortical Anatomy. In *Proc. of SPIE Visualization in Biomedical Computing*, volume 2359, pages 160–169, March 1994.

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

[30] T. McInerney and D. Terzopoulos. Medical Image Segmentation Using Topologically Adaptable Surfaces. In J. Troccaz, E. Grimson, and R. Mösges, editors, *Proc. of CVRMed-MRCAS*, volume 1205 of *Lecture Notes in Computer Science*, pages 23–32, Grenoble, France, March 1997. Springer-Verlag.

[31] 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 (SIGGRAPH'91 Proceedings)*, 25(4), July 1991.

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

[33] J. Montagnat and H. Delingette. Volumetric Medical Images Segmentation Using Shape Constrained Deformable Models. In J. Troccaz, E. Grimson, and R. Mösges, editors, *Proc. of CVRMed-MRCAS*, volume 1205 of *Lecture Notes in Computer Science*, pages 13–22, Grenoble, France, March 1997. Springer-Verlag.

[34] J. Park, D. Metaxas, and L. Axel. Analysis of left ventricular wall motion based on volumetric deformable models and MRI-SPAMM. *Medical Image Analysis*, 1(1):53–71, March 1996.

[35] S. Peleg and O. Federbusch. Custom Made Pyramids. In V. Cantoni and S. Leviadi, editors, *Pyramidal Systems for Computer Vision*, volume F 25 of *NATO ASI Series*, pages 165–171. Springer-Verlag, Berlin Heidelberg, 1986.

[36] E. Promayon, P. Baconnier, and C. Puech. Physically-Based Deformations Constrained in Displacements and Volume. *Computer Graphics forum*, 15(3):155–164, 1996.

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

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

[39] P. Volino and N.M. Thalmann. Efficient self-collision detection on smoothly discretized surface animations using geometrical shape regularity. In M. Dæhlen and L. Kjelldahl, editors, *Proc. of Eurographics'94*. Blackwell Publishers, 1994.

[40] R.T. Whitaker. Volumetric deformable models: active blobs. In R.A. Robb, editor, *Proc. Third Conf. on Visualization in Biomedical Computing (VBC'94)*, volume 2359 of *SPIE Proc.*, pages 122–134, Rochester, MN, October 1994. SPIE.

[41] E. Wolfson and E.L. Schwartz. Computing minimal distances on polyhedral surfaces. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 11(9):1001–1005, 1989.

[42] J.C. Xia, J. El-Sana, and A. Varshney. Adaptive Real-Time Level-of-Detail-Based Rendering for Polygonal Models. *IEEE Trans. on Visualization and Computer Graphics*, 3(2):171–183, April 1997.

A Construction of a three-dimensional pyramid

Let I be an image of discrete sizes $M \times N \times P$ and of real sizes $\mu \times \nu \times \pi$. ρ is the reduction factor and ρ_{max} the maximal reduction. The following procedure *BuildPyramid*() builds a pyramid P from the image I with a reduction ρ.

```
Procedure BuildPyramid ( const Image & I, PyramidOfImage & P, double ρ )
    int h ⟵ 0
    int Mupper, Nupper, Pupper
    int Mlower ⟵ I.M
    int Nlower ⟵ I.N
    int Plower ⟵ I.P
    P.image(h) ⟵ I
    while ρ^h < ρ_max do
        int Mupper ⟵ ⌊Mlower/ρ⌋
        int Nupper ⟵ ⌊Nlower/ρ⌋
        int Pupper ⟵ ⌊Plower/ρ⌋
        Image G(Mupper, Nlower, Plower)
        G ⟵ ConvolutionAlongX ( P.image(h), [1 4 6 4 1]/16, ρ)
        Image H(Mupper, Nupper, Plower)
        H ⟵ ConvolutionAlongY ( G, [1 4 6 4 1]/16, ρ)
        Image P.image(h + 1) (Mupper, Nlower, Pupper)
        P.image(h + 1) ⟵ ConvolutionAlongZ( H, [1 4 6 4 1]/16, ρ )
        delete G, H
        h ⟵ h − 1
    done
end
```