The Marching lines algorithm : new results and proofs

Jean-Philippe Thirion, Alexis Gourdon

▶ **To cite this version:**

Jean-Philippe Thirion, Alexis Gourdon. The Marching lines algorithm : new results and proofs. [Research Report] RR-1881, INRIA. 1993. inria-00074792

HAL Id: inria-00074792
https://inria.hal.science/inria-00074792

Submitted on 24 May 2006

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

The marching lines algorithm : new results and proofs

Jean-Philippe THIRION
Alexis GOURDON

N° 1881
Avril 1993

PROGRAMME 4

Robotique, Image
et
Vision

The Marching Lines Algorithm : new results and proofs

Jean-Philippe THIRION and Alexis GOURDON
INRIA, Equipe Epidaure
B.P. 93, 2004 route des Lucioles
06902 SOPHIA ANTIPOLIS Cedex, FRANCE
Email: jean-philippe.thirion@inria.fr
Tel: (33) 93 65 76 62
Fax: (33) 93 65 79 69

Abstract

This research report is a compilation of two articles describing new results concerning the 3D Marching Lines algorithm. The Marching Lines extracts, with sub-voxel accuracy, characteristic 3D lines out of 3D images, such as the Crest Lines. Those feature lines can then be used to perform higher level Image Processing tasks, such as 3D image registration, or automatic labeling of anatomical structures, when Medical Images are processed. The first paper concentrates on the computation of the differential characteristics of iso-intensity surfaces, and shows how to characterize Crest Lines points directly from the differentials of the 3D Image. The second paper brings the proof of the good topological properties of the reconstructed surfaces and 3D curves obtained with the Marching Lines algorithm. New experiments on real and synthetic data are also presented, showing the high precision and stability of the extracted feature lines.

Key words: 3D Image, Iso surface, Crest Line, Topology, Differential Geometry, Marching Lines.

L'algorithme des "Marching Lines", nouveaux résultats et preuves

Jean-Philippe THIRION and Alexis GOURDON
INRIA, Equipe Epidaure
B.P. 93, 2004 route des Lucioles
06902 SOPHIA ANTIPOLIS Cedex, FRANCE
Email: jean-philippe.thirion@inria.fr
Tel: (33) 93 65 76 62
Fax: (33) 93 65 79 69

Résumé

Ce rapport de recherche est la compilation de deux articles décrivant de nouveaux résultats concernant l'algorithme des "Marching Lines". Cet algorithme permet d'extraire, avec une précision inférieure à la taille des éléments de volumes, et à partir d'images tri-dimensionnelles, des lignes caractéristiques 3D, telles que les lignes de crêtes, par exemple. Ces lignes remarquables peuvent être ensuite utilisées pour réaliser des tâches de traitement d'images de plut haut niveau, tel que le recalage d'images 3D, ou l'étiquetage automatique de structures anatomiques, dans le cas des images médicales. Le premier article se concentre sur le calcul des caractéristiques différentielles des surfaces d'iso-intensité, et montre comment caractériser les points des lignes de crête, directement à partir des différentielles de l'image 3D. Le second article apporte la preuve que les bonnes propriétés topologiques des iso-surfaces et des lignes caractéristiques, reconstruites par notre algorithme des "Marching Lines", sont bien respectées. Nous présentons de nouveaux résultats expérimentaux, qui démontrent la très haute précision et la stabilité des lignes remarquables extraites par notre algorithme.

Mots clefs: images 3D, iso-surface, ligne de crête, topologie, géométrie différentielle, Marching Lines.

Computing the Differential Characteristics of Iso-Intensity Surfaces

Jean-Philippe THIRION and Alexis GOURDON
INRIA,
B.P. 93, 2004.route des Lucioles
06902 SOPHIA ANTIPOLIS Cedex, FRANCE
Email: jean-philippe.thirion@inria.fr
Tel: (33) 93 65 76 62
Fax: (33) 93 65 79 69

Abstract

In this paper, we present a new method to compute the differential characteristics of iso-intensity surfaces from three-dimensional images. We show applications where those differentials properties are used to extract characteristic lines from 3D images, called crest lines. The crest lines are then used to perform the registration of 3D images with a precision that demonstrates the reliability and robustness of our computation. Those experiments also show the direct correspondence between geometrical and anatomical features, for medical images. To compute the differential characteristics of surfaces, such as the principal curvatures, and directions, a traditional approach is to fit a parametric surface model to the 3D image, and then to compute the differential characteristics of the surface in the local coordinate system. On the contrary, our method is based on the implicit representation of the surface. We can compute directly the differential values of the iso-intensity surfaces from the 3D image, *without* extracting any surface first. In our method, the principal curvatures and directions equations have been derived from the implicit functions theorem, leading to entirely new formulae, which make use of only the differentials of the 3D image, and which allow us to get rid of the problem of parametrizing the surfaces.

Key words: 3D Image Processing, Differential Geometry, Iso-Surfaces, Crest Lines, Marching Lines.

1 Introduction

Three dimensional image processing is of growing interest in the fields of image processing and computer vision. To see how to extend existing 2D image processing techniques to 3D gives new and unexpected insights into methods which were thought to be well known and studied. Sometimes, transforming the 2D method into 3D is straightforward, but in many cases, the problem has to be entirely reconsidered. In many occasions, finding a way to extend the solution from 2D to 3D is a way to extend it to ND. Sometimes also, the proposed algorithms are tradeoffs between cpu time and memory requirement. Due to the enormous amount of information corresponding to the 3D images, those tradeoffs change, and entirely new methods have to be designed.

This is the case for the method that we present in this paper. As we will see, it is the extension to 3D of an existing algorithm of 2D image processing, used to extract corner points from 2D images, but it is not a straightforward extension, because our method relies on results of the 3D differential geometry, which is far more different and complex than the 2D case.

The problem is how to compute the differential characteristics of the surfaces extracted from 3D images. Those images are not depth images or range images, but are really three dimensional images composed with almost isotropic voxels, such as the compilation of Xray scanner slices, or 3D Magnetic Resonance images (MRI). Those surfaces can be for example the surface of organs, such has the surface of the skin, the skull, the brain, etc .... The motivation can be the registration of two organs, in order to compare them, based on the matching of points of the surfaces which have the same differential characteristics, for example similar principal curvatures. Principal curvatures, or gaussian and average curvature, are actually invariant with respect to 3D rigid transforms, and also invariant to some classes of deformations. Particularly, the crest lines, which are the lines corresponding to the successive loci of the surface whose principal curvature is locally maximal, have proved to be reliable landmarks to perform the registration of 3D images (see [14], [15]). What is absolutely remarkable is that those crest lines, that we can extract automatically from the 3D images, and whose extraction is based only on geometrical features, happen to be anatomical features in the case of medical images. For, example, in the case of the skull, those crest lines are the eyes and nose orbital lines, or the sub-mandibular line. In the case of the brain, these are called "Gyri" and "Sulci" by the physicians, corresponding respectively to the crests and the groves of the circumvolution of the brain. For those who are not entirely convinced by the usefulness of the differential geometry features, those two kinds of lines (gyri or sulci), can be distinguished with the sign of the maximal curvature.

We first present previous works about the computation of the differential characteristics of surfaces, and explain how our method differs from them. Then we introduce some definitions, such as the differential characteristics of the surfaces, or the definition of the iso-intensity surfaces. We then describe

the restriction of our method to the 2D case. This will be useful to fully understand the next section, the 3D case, which constitute the core of this paper. At last, we will show experimental results for synthetic and real data, and particularly the extraction of crest lines, which is automatically performed with the use of an algorithm called the "Marching Lines", described in [14].

2 Related works

The differential geometry of 3D surfaces has been extensively studied since a long time, and, as early as the end of the 19th century, the major results were already known. Many books can be found, describing those results, such as [4], or [9]. The interest about the differential characteristics of surfaces in the field of computer vision and image processing appeared since range data, also called depth images, were provided (see for example [2]). The surface corresponding to those " two and a half" dimension data can be easely parametrized, using the natural coordinates of the 2D image $f(x, y)$ representing those data. There is an exact correspondence between the coordinates (x, y) of the image, and the local coordinates of the surface $z = f(x, y)$. The classical formulae of differential geometry can be directly applied, because most of those results rely on the *parametric* representation of the surfaces.

The major problem was to extend this work to surfaces extracted from real 3D images. Beside the problem of the segmentation, the topology of those surfaces can be very complex, which leads to the tremendous problems of the parametrization of those surfaces. A natural parametrization of the surfaces is the use of the lines of curvature (see [4]), but unfortunately, one has to know the differential characteristics of the surface to perform this parametrization, and it is exactly what we are looking for. Almost all the methods which have been proposed (see [11], [1], [5]) were to fit locally a surface patch to the data, with a known parametrization in a local coordinate system (u, v), and to rely then once again on the traditional results of differential geometry. The major problem of those methods is how to fit (and sometime deform) the surface patch in a reliable way, which is independent from the topology of the surface.

On the contrary of all these methods, there is a remarkable and fundamental work performed by Monga, Benayoun and Faugeras in [10]. The major contribution of their work was to show that it was possible to compute locally the gaussian, average, and principal curvatures, and also the principal directions, without fitting a surface patch. They use the gradient direction $\vec{g}$ to define the tangent plane of the surface. Then they define a local parametrization by giving two arbitrary perpendicular vectors $\vec{a}, \vec{b}$ in this plane. To find the principal directions, they define a unit vector $\vec{t}$ in the $\vec{a}, \vec{b}$ plane, characterized by its angle θ around the normal $\vec{g}$. They have shown how to compute the curvature $k_{\vec{t}}$ of the surface in the direction $\vec{t}(\theta)$ by using the Hessian matrix H of the 3D image function at this point (see equation 1), and how to compute the principal directions, which are the directions where

the curvature of the surface is extremal, by a search of the extrema of $k_{\vec{t}}(\theta)$.

$$k_{\vec{t}} = -\frac{t^{\perp} H t}{||\vec{g}||} \tag{1}$$

At last, they also give a characterization of the crest line points, being the zero-crossings of a criterion that they called the "extremality function", which is the directional derivative of the maximal curvature, in the associated principal direction.

Our method is close to this last work, in that we compute the differential characteristics, and also the extremality function of the surface, directly from the voxel values : more precisely from the differentials of the image function $f(x, y, z)$. The major difference with this last work is that we do not consider a parametric representation of the surface, but its *implicit* definition, and that we derive our solutions from the application of the implicit functions theorem. Our final formulae use only the differentials of the image function $f(x, y, z)$ in the three directions $\vec{x}, \vec{y}, \vec{z}$, and are symmetrical with respect to those axes. At last, what is noticeable is that the differential characteristics of the surface can be computed at any point P of the 3D image, if the gradient norm is not zero. The interpretation of this is that the computed values at point P corresponds the differential characteristics of the iso-intensity surface coming through the point P, that is, the iso-surface whose iso-value is the intensity at the point P (if not familiar with the notion of iso-surfaces, see the next section.)

3 Definitions

We recall in this section some definitions and properties of the 3D surfaces, that will be useful in the next sections.

First, let us define the iso-intensity surface of a 3D image, which will be called simply the iso-surface in the rest of this paper. For any continuous function $f(x, y, z)$ of $\mathcal{R}^3$, any value I of $\mathcal{R}$ (called the iso-value) defines a continuous, not self intersecting surface, without hole, which is called the iso-intensity surface of f. A not ambiguous way to define the iso-surface is to define it as being the surface which separates the regions of the space where the intensity of f is greater or equal to I from the region whose intensity is strictly lower than I (see figure 1). One can see that we could have chosen also the surface which separates the regions $f > I$ from regions $f \leq I$. The two surfaces are the same, except at the boundary of the constant regions. In fact, this choice is exactly the same as the choice which has to be made for the discrete representation of the image, for example in 2D, when choosing between the 4 and 8 connexity. One has to give some privilege to either the object, or to the background, and this has to be done for the discrete case as for the continuous case. The surface cannot simply be defined by the implicit equation $f(x, y, z) = I$, because some regions of $\mathcal{R}^3$ may have a constant intensity I, and in that case, this formula does not define a surface, but a volume. But, except for the boundary of the constant regions (where

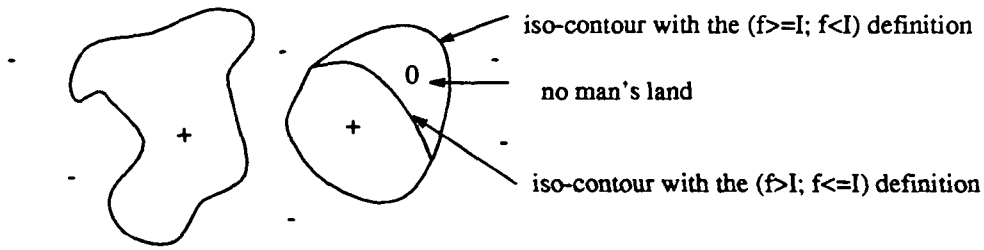


Figure 1: Definition of iso-contours

the gradient is null), the implicit equation $f(x, y, z) = I$ of the iso-surface holds. That this iso-surface corresponds to the boundary of the scanned object is another problem. In many cases, such as for medical images, the iso-surface techniques are directly used to segment the organs, for example the bones in CT scans (see figure 9). Sometimes however, a pre-processing step is necessary to determine which part of the image is the object, and which part is the background. This pre-segmentation, composed with smoothing filters, morphological operators and search of connex components, leads to a 3D image where the intensity roughly represent the probability of being inside or outside the object. Iso-surface techniques are then used in the final phase to extract the surface, as it is the case for the segmented brain presented in the figure 12. They ensure that the reconstructed surfaces are continuous, not self-intersecting, and without hole (except of course for the image boundary). Because of those good topological properties, iso-surface techniques are the most widely used methods of segmentation for 3D medical images.

Let us see now some properties of the two times differentiable surfaces (see figure 2). At each point P of those surfaces, there is an infinite number of curvatures, but, for each direction $\vec{t}$ in the tangent plane at P, there is only one associated curvature $k_{\vec{t}}$. There are two privileged directions of the surface, called the principal directions ($\vec{t_1}$ and $\vec{t_2}$), which correspond to the two extremal values of the curvature : k_1 and k_2 (except for the umbilic points, for which the curvature is the same for all the directions). One of these two principal curvatures is maximal in absolute value (let say k_1), and is called the maximal curvature. A noticeable result is that the knowledge of the two principal curvatures and directions suffices to determine any other curvature at point P. The product of the two principal curvatures is called the gaussian curvature K, an the half sum is called the average curvature S. Those differential values: $\vec{t_1}$, $\vec{t_2}$, k_1, k_2, K and S, can be used in many different ways to characterize locally the surface. To those values, we can add the extremality criterion e as defined by Monga et al. in [10], which is the directional derivative of the maximal curvature (let say that it is k_1), in the corresponding principal direction ($\vec{t_1}$). In fact, the same extremality criterion can be also defined for the other principal direction, and we have therefore

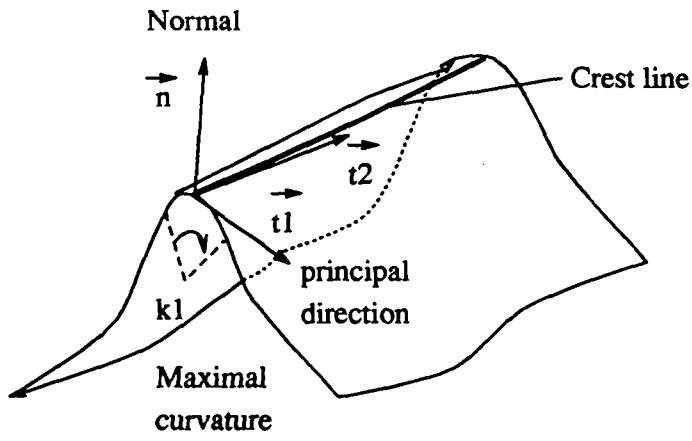


Figure 2: Differential characteristics of surfaces

two "extremalities" e_1 and e_2. The successive loci of the zero-crossing of the extremality criterion define lines, which are called ridge lines in [10], but sometime also crest lines.

Works about crest lines are very rare in the traditional differential geometry bibliography. In fact, there is a problem in this last definition, which comes from the fact that $\vec{t_1}$ and $\vec{t_2}$ are not defined as vectors, but as directions (and sometimes are only noted t_1 and t_2). We take by definition that $\vec{t_1}$, $\vec{t_2}$ and the normal of the surface form a direct trihedron, that is : $\vec{t_1} \wedge \vec{t_2} = \vec{\nabla} f / ||\vec{\nabla} f||$, where $\vec{\nabla} f$ is the gradient of f, and therefore, is the normal of the iso-surface. But either $(\vec{t_1}, \vec{t_2})$ or $(-\vec{t_1}, -\vec{t_2})$ are still valid sets of principal direction vectors. The use of the Marching Lines algorithm, as defined in [14] is a way to avoid that problem, by orienting locally the principal directions, according to the direction of the tangent of the line which is currently extracted. This point will be discussed later in this paper.

4 Curvature and corner points in 2D images

In this section, we present the restriction of our method to the 2D case, which will help us a lot to understand the computations of the next section, for the 3D case. In fact, similar works have already been performed in 2D (see [8], [12], [6]), in the field of computer vision, for the detection of corner points, which correspond to the "corners" or "wedges" of the geometric objects. In fact, except for the use of our extremality coefficient, this part leads to similar results as in [8].

For a 2D function $f(x, y)$ of $\mathcal{R}^\epsilon$, we call "iso-boundary" the restriction of the iso-intensity surface to 2D: the iso-boundaries are the lines which separate the regions $f \geq I$ from the regions $f < I$. We show now how to compute the curvature of this iso-contour, and how to derive a 2D extremality criterion e whose zero-crossings are the points of maximal curvature along the iso-

boundary.

Of course, an image i is only a digital representation of a continuous function. We will assume that $f(x,y)$ is the continuous function obtained with the convolution of the image i with the gaussian function, therefore f is continuous and infinitely differentiable (of course, other interpolation functions can be studied). The iso-boundary defined with the iso-value I is also infinitely differentiable and its equation is $f(x,y) = I$, if $||\vec{\nabla} f|| \neq 0$. This equation is the implicit equation of a curve in the plane. Using the implicit functions theorem, there exists locally a function ϕ such that $(x = u, y = \phi(u))$ and $f(u, \phi(u)) = I$. The derivative of ϕ satisfies:

$$\frac{d\phi}{du}(u) = -\frac{\partial f(u,\phi(u))/\partial x}{\partial f(u,\phi(u))/\partial y} \tag{2}$$

We will use the following traditional notations for the partial derivatives: f_x for $\partial f(x,y)/\partial x$, f_{xy} for $\partial^2 f(x,y)/\partial x \partial y$ etc., and ϕ' for $d\phi/du$. We thus have:

$$\phi' = -f_x/f_y \tag{3}$$

Replacing ϕ by ϕ' in equation 2 gives ϕ'':

$$\phi'' = \frac{2f_x f_y f_{xy} - f_x^2 f_{yy} - f_y^2 f_{xx}}{f_y^3} \tag{4}$$

The curvature c of the curve is given by $c(u) = \phi''(u)/(1 + \phi'^2(u))^{3/2}$, therefore:

$$c(x,y) = \frac{2f_x f_y f_{xy} - f_x^2 f_{yy} - f_y^2 f_{xx}}{(f_x^2 + f_y^2)^{3/2}} \tag{5}$$

Which is the equation that give the curvature at any point of the image, and which depends only on the differentials of the image, up to order 2. Let see now how to compute the points of maximum curvature along the iso-boundary. Those points satisfies the equation $dc(u)/du = 0$. A strictly equivalent equation is to state that the derivative of the curvature in the direction of the tangent is zero, that is $\vec{\nabla} c(x,y) \cdot \vec{t} = 0$, where $\vec{t}$ is the tangent $(-f_y, f_x)$ to the iso-boundary, and $\vec{\nabla} c(x,y)$ is the gradient of c with respect to x and y: (c_x, c_y). With this notation:

$$c_x f_y - c_y f_x = 0 \tag{6}$$

Which can be easily expanded using Equation 5. Equation 6 is of the form $e(x,y) = 0$ and defines a new implicit curve in the 2D image, whose intersection with the iso-boundary $f(x,y) = I$ are the points of maximal or minimal curvatures. We call the curve $e(x,y) = 0$ the *maximal (resp. minimal) curvature curve* (see figure 3). It corresponds to the loci of the maximum curvature points for a continuous variation of the iso-value I. This explains why $e(x,y)$ is defined everywhere, when the gradient is not null, because at any point (x,y), there exists an iso-contour whose iso-value is

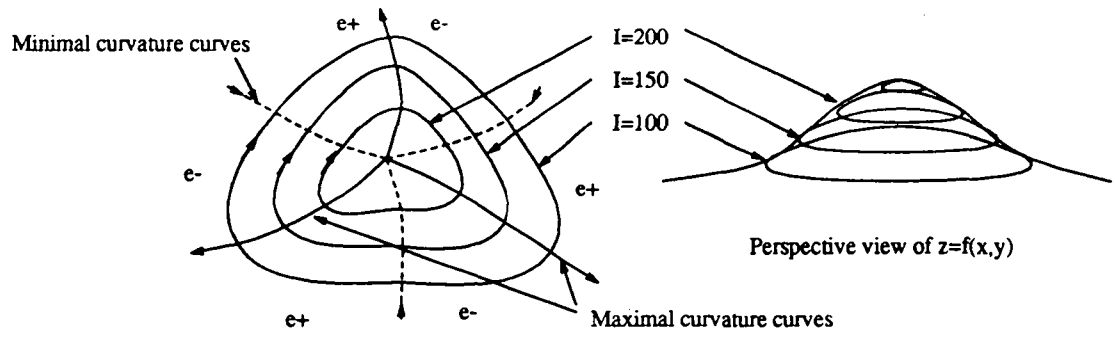


Figure 3: Maximal and minimal curvature curves

$f(x, y)$. What is remarkable is that the maximum curvature curve does not depend on the choice of any iso-value threshold I, but is intrinsically defined with the values of the image $f(x, y)$. Of course, the extremality is really significant only in the neighborhood of a "true" contour, that is, when the gradient norm is high.

Figure 4 represents a zoom in an image, for which we have computed both the iso-contour and the maximal curvature curve. It is a Cat scan image at the level of the ear, which is a particularly difficult image for the detection of corner points, since the corners in this kind of image are far more smoothed that for traditional computer vision images, displaying polyhedral objects. The intersection of those two lines sets are the corner points. We can see on this figure the influence of the smoothing factor σ of the gaussian filter, which is comparable with a scale factor. Perhaps a multi-scale approach would help to isolate "true" corner points from those only due to noise, the relevent corner points being those which are stable with respect to scale (i.e. σ). In the case of crest lines, we will see that the length of the extracted crest lines can be used to efficiently determine the relevent crest points.

In order to have reliable results, we have to compute the differentials of the image with the use of a large σ, and therefore, with a wide kernel. Fortunately, the gaussian filter can be implemented in a separated way, and is not too much time consuming, even for a large support and for 3D images. Furthermore, there exist a recursive implementation of a good approximation of the gaussian function, which is even faster, and works in 3D (see [3]).

As we can see, we have reduced the determination of the corner points to the computation of the intersection of two planar curves $f(x, y) = I$ and $e(x, y) = 0$. Let us see now the 3D case. We will show that the computation of crest lines on iso-intensity surfaces can be reduced to the computation of the intersection of two iso-surfaces, one being the classical iso-surface $f(x, y, z) = I$, and the other one being the zero-crossing of an extremality function $e_1(x, y, z) = 0$. We call this last implicit surface, the crest surface.

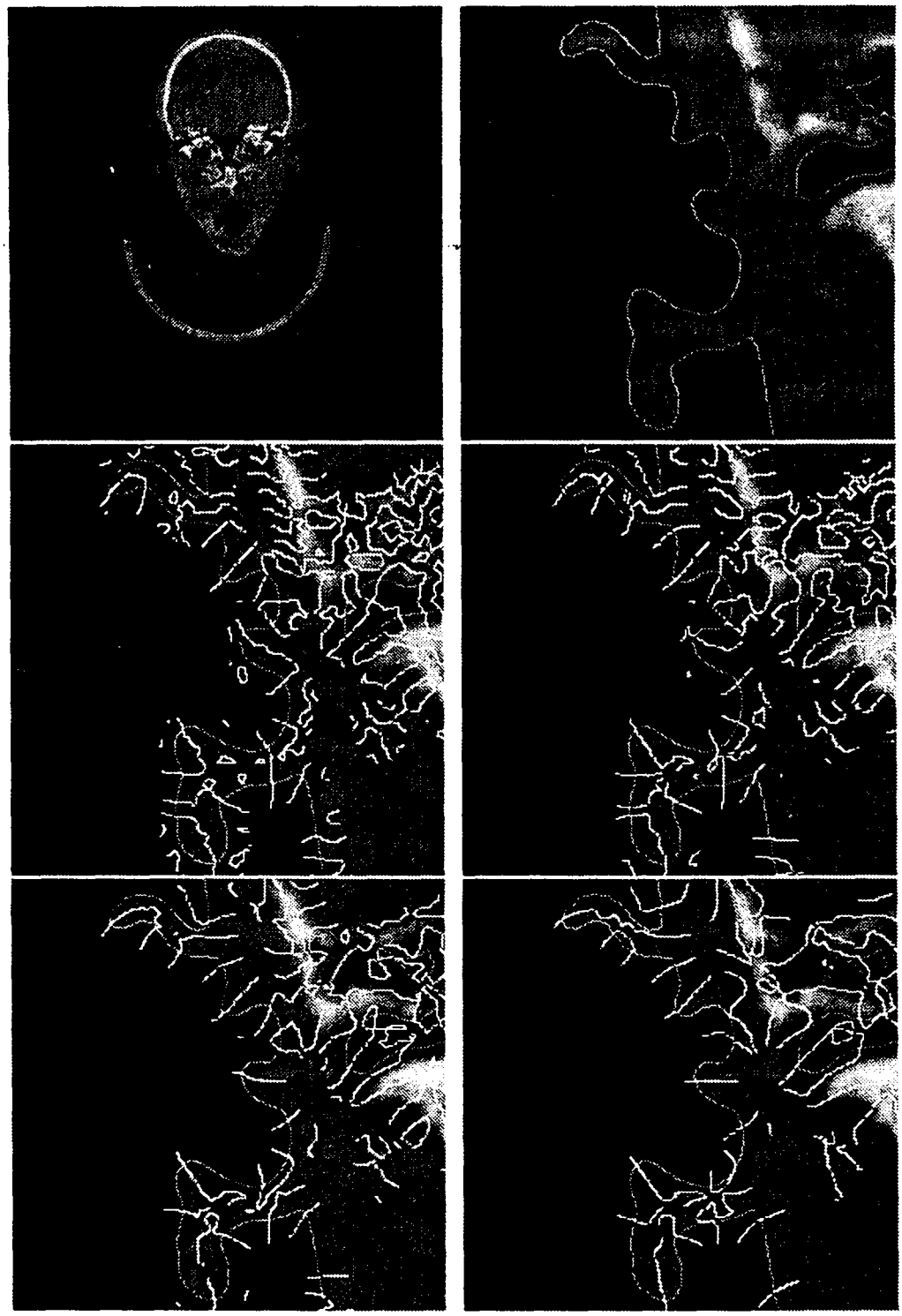


Figure 4: Original image, zoom displayed with an iso-contour, and computed maximal curvature curves for $\sigma = 1$, 1.5, 2 and 3 (threshold with the gradient value).

5 The differential characteristics of iso-surfaces.

Once again, we use the convolution with the gaussian function to transform the digital 3D image into an infinitely differentiable 3D function $f(x, y, z)$. As for the 2D case, we use the implicit functions theorem in order to transform the implicit equation of the iso-surface $f(x, y, z) = I$ of f into the parametric equation of the surface: $(x = u, y = v, z = \phi(u, v))$. Then the first and second Fundamental Forms of differential geometry (see for example [4]) can be used to compute the principal curvatures and principal directions. These formulae are generally written for the parametric formulation. The contribution of our paper is to find equivalent results with the implicit equation of a surface. The principal curvatures and principal directions correspond respectively to the eigenvalues and the eigenvectors of the matrix, called the Weingarten endomorphism :

$$\begin{pmatrix} E & F \\ F & G \end{pmatrix}^{-1} \begin{pmatrix} L & M \\ M & N \end{pmatrix} = \begin{pmatrix} \frac{(GL-FM)}{H} & \frac{(GM-FN)}{H} \\ \frac{(EM-FL)}{H} & \frac{(EN-FM)}{H} \end{pmatrix} = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \quad (7)$$

The coefficients are computed as follow. S is the parametric surface defined with $S(u, v) = (u, v, \phi(u, v))$. For a point $S(u, v)$ of this surface, we have :

$$E = \|\vec{S}_u\| \quad F = \vec{S}_u \cdot \vec{S}_v \quad G = \|\vec{S}_v\| \quad (8)$$

$$L = \frac{\vec{S}_{uu} \cdot \vec{n}}{H^{1/2}} \quad M = \frac{\vec{S}_{uv} \cdot \vec{n}}{H^{1/2}} \quad N = \frac{\vec{S}_{vv} \cdot \vec{n}}{H^{1/2}} \quad and \quad H = EG - F^2 \quad (9)$$

Here, $\vec{n}(u, v)$ is defined as $\vec{S}_u \wedge \vec{S}_v$, and is a vector lying normal to the surface at $\vec{S}(u, v)$. Using the implicit functions theorem, and substitution of the derivatives of f for derivatives of $\vec{S}$, as in the 2D case:

$$\begin{array}{lll} E = \frac{f_x^2+f_z^2}{f_z^2} & L = \frac{2f_x f_z f_{xz} - f_x^2 f_{zz} - f_z^2 f_{xx}}{H^{1/2} f_z^3} & \\ F = \frac{f_x f_y}{f_z^2} & M = \frac{f_x f_z f_{yz} + f_y f_z f_{xz} - f_x f_y f_{zz} - f_z^2 f_{xy}}{H^{1/2} f_z^3} & H = \frac{f_x^2+f_y^2+f_z^2}{f_z^2} \\ G = \frac{f_y^2+f_z^2}{f_z^2} & N = \frac{2f_y f_z f_{yz} - f_y^2 f_{zz} - f_z^2 f_{yy}}{H^{1/2} f_z^3} & \end{array} \quad (10)$$

The formulae are not symmetric with respect to the three principal axes $\vec{x}, \vec{y}, \vec{z}$, because we gave a special role to $\vec{z}$ when we parametrized the equation by the implicit functions theorem. We are looking for symmetric equations for the curvatures and principal directions, let see how this can be achieved : k_1, k_2 are the two principal curvatures and $\vec{t_1}, \vec{t_2}$ are the associated principal directions. $K = k_1 k_2$ is the gaussian curvature and $S = (k_1 + k_2)/2$ is the average curvature of the iso-surface. K and S are also respectively the determinant and half the trace of the matrix given in Equation 7, that is:

$$K = (LN - M^2)/H \quad and \quad 2S = (EN - 2FM + GL)/H \quad (11)$$

Using Equation 10:

$$K = \frac{1}{h^2}[\begin{array}{ll} f_x^2(f_{yy}f_{zz} - f_{yz}^2) + & 2f_yf_z(f_{xz}f_{xy} - f_{xx}f_{yz}) + \\ f_y^2(f_{xx}f_{zz} - f_{xz}^2) + & 2f_xf_z(f_{yz}f_{xy} - f_{yy}f_{xz}) + \\ f_z^2(f_{xx}f_{yy} - f_{xy}^2) + & 2f_xf_y(f_{xz}f_{yz} - f_{zz}f_{xy})] \end{array} \quad (12)$$

$$S = \frac{1}{2h^{3/2}}[\begin{array}{ll} f_x^2(f_{yy} + f_{zz}) - & 2f_yf_zf_{yz} + \\ f_y^2(f_{xx} + f_{zz}) - & 2f_xf_zf_{xz} + \\ f_z^2(f_{xx} + f_{yy}) - & 2f_xf_yf_{xy}] \end{array} \quad (13)$$

where $h = f_x^2 + f_y^2 + f_z^2$. $\vec{x}, \vec{y}, \vec{z}$ have the same role in these equations. Once we have found K and S, k_1 and k_2 are the solutions of an equation of order two:

$$k_i = S \pm \sqrt{\Delta} \quad with \quad \Delta = S^2 - K. \quad (14)$$

This demonstrates that the principal curvatures of the iso-surfaces can be computed directly from the first and second derivatives of the image. Our formulae do not give a specific play to a particular direction of the space.

The computation of the principal directions, however, is a bit more complicated. Using the notation of Equation 7, the principal directions $\vec{t_i}$ (for $\vec{t_1}, \vec{t_2}$), which are the two eigenvectors of the matrix in Equation 7, may be represented in the basis $\vec{S_u}, \vec{S_v}$ in the form $u_1\vec{S_u} + v_1\vec{S_v}$, and satisfy the equation:

$$\begin{array}{rcrcl} (a - k_i)u_i & + & bv_i & = & 0 \\ cu_i & + & (d - k_i)v_i & = & 0 \end{array} \quad (15)$$

For each $i \in 1, 2$, the two equations are dependent, and thus the solution set for (u_i, v_i) will lie along a line colinear with $\vec{t_i}$. We can compute $\vec{t_i}$ with either of the two equations of 15, which give us two vectors $\vec{t_{i_1}}$ and $\vec{t_{i_2}}$ colinear to $\vec{t_i}$. Since $\vec{S_u} = (1, 0, -f_x/f_z)$ and $\vec{S_v} = (0, 1, -f_y/f_z)$ we have:

$$\vec{t_{i_1}} = \begin{pmatrix} f_zb \\ f_z(k_i - a) \\ -f_xb - f_y(k_i - a) \end{pmatrix} \quad \vec{t_{i_2}} = \begin{pmatrix} f_z(k_i - d) \\ f_zc \\ -f_x(k_i - d) - f_yc \end{pmatrix} \quad (16)$$

Some simplifications are possible, we have:

$$(k_i - a) = \frac{EN - GL}{2H} \pm \sqrt{\Delta} \quad , \quad (k_i - d) = \frac{GL - EN}{2H} \pm \sqrt{\Delta} \quad (17)$$

We are now at a crucial point of the paper. Once again, we are looking for symmetric equations in x, y, z. We found a solution, which is to use a given linear combination of $\vec{t_{i_1}}$ and $\vec{t_{i_2}}$. Making the substitutions for $E, N, G, L,,$ and then using the linear combination $\vec{t_i} = \frac{f_z - f_z}{f_z}\vec{t_{i_1}} + \frac{f_z - f_y}{f_z}\vec{t_{i_2}}$ restore the symmetry between the three coordinates. We obtain:

$$\vec{t_i} = \vec{\alpha} \pm \sqrt{\Delta}\vec{\beta} \quad with \quad \vec{\beta} = (f_z - f_y, f_x - f_z, f_y - f_x) \quad (18)$$

The vector $\vec{\alpha}$ is more complicated. It depends on the first and second derivatives of the image function, and, like $\vec{\beta}$, it is symmetric with respect

to the three coordinates. Equation 19 gives the components of vector $\vec{\alpha}$. The y and z components are obtained by circular permutations of x, y and z. Despite the symmetry between the three axes, there is still a privileged direction, which is $\vec{w} = (1,1,1)$, because, if $\vec{\nabla} f$ is colinear to $\vec{w}$, then $\vec{\alpha} = \vec{\beta} = 0$, which means that we have failed to find $\vec{t_1}, \vec{t_2}$ in that case. There are thus locations where the principal directions are not obtained by means of the symmetric formula : for umbilic points ($\Delta = 0$), when the gradient vanishes ($\vec{\nabla} f = 0$), which was expected, but also for a privileged direction ($\vec{\nabla} f \wedge \vec{w} = 0$), which is troublesome, an rather unexpected.

$$\begin{aligned}\vec{\alpha}\cdot\vec{x} = -\tfrac{1}{2h^{3/2}}[\; & -2f_z^3 f_{xy} & +f_y^3 f_{zz} & +2f_y^3 f_{xz} & -2f_y^2 f_z f_{xy} \\ & +2f_z^2 f_x f_{yz} & +2f_z^2 f_y f_{xz} & -2f_y^2 f_x f_{yz} & -2f_z f_x f_y f_{zz} \\ & +2f_x f_y f_z f_{yy} & +f_y^2 f_z f_{xx} & -2f_z^2 f_x f_{xz} & +f_z f_x^2 f_{zz} \\ & -f_x^2 f_z f_{yy} & +2f_z^2 f_y f_{yz} & -f_z f_y^2 f_{zz} & +f_z^3 f_{xx} \\ & -f_z^3 f_{yy} & -2f_y^2 f_x f_{xz} & +2f_x^2 f_y f_{yz} & -f_y^3 f_{xx} \\ & +2f_x f_z^2 f_{xy} & -f_y f_z^2 f_{xx} & -2f_z f_y^2 f_{yz} & +f_y f_z^2 f_{yy} \\ & -2f_z f_x^2 f_{yz} & +2f_x f_y^2 f_{xy} & +f_x^2 f_y f_{zz} & -f_x^2 f_y f_{yy}]\end{aligned} \tag{19}$$

If k_1 is the maximum curvature in absolute value ($|k_1| \geq |k_2|$), then the gradient of k_1 in the direction of $\vec{t_1}$ is given by $e_1 = \vec{\nabla} k_1 \cdot \vec{t_1}$, which can be computed from Equations 14 and 18. Setting the directional derivative to zero, we obtain a *"crest surface"* $e_1(x,y,z) = 0$, containing points where the value of k_1 is extremal in the $\vec{t_1}$ direction. The intersection of the crest surface with the iso-surface of the image gives the crest lines (see figure 5). The Marching Lines algorithm (see [14]) can then be used to automatically extract those crest lines. In fact, these reconstructed curves are not necessarily closed because there are some points where $\vec{t_1}$ is not defined, and therefore e_1 is not defined. In that case, one simple solution is to stop the "marching" when e_1 is not defined. Furthermore, only some parts of the extracted lines may be considered as valid crest lines: the "quality" of a point being a crest point can be estimated from the absolute value of the gradient (which indicates whether the point of the iso-surface belongs to the boundary of an object or not) and the absolute value of the curvature. We found experimentally that a simple threshold on the length of the extracted lines suffices to separate the significant crest lines, from those only due to noise.

The important point is that we have reduced the study of a 3D image to a 1D variety of points: the 3D lines defined with the intersection of the iso-surfaces $f(x,y,z) = I$ and the crest surface $e_1(x,y,z) = 0$.

5.1 Why the crest surface is only locally defined ?

A close look at the computation of the principal directions reveals that the orientation of those directions is meaningless (i.e. either $(\vec{t_1}, \vec{t_2})$ or $(-\vec{t_1}, -\vec{t_2})$ are valid principal directions). As we saw, even the symmetric formula corresponds to a specific orientation of the space. In fact, there is no globally continuous solution that fixes an orientation of the principal directions over a closed surface. The reason is the same as a result of geometry, sometimes

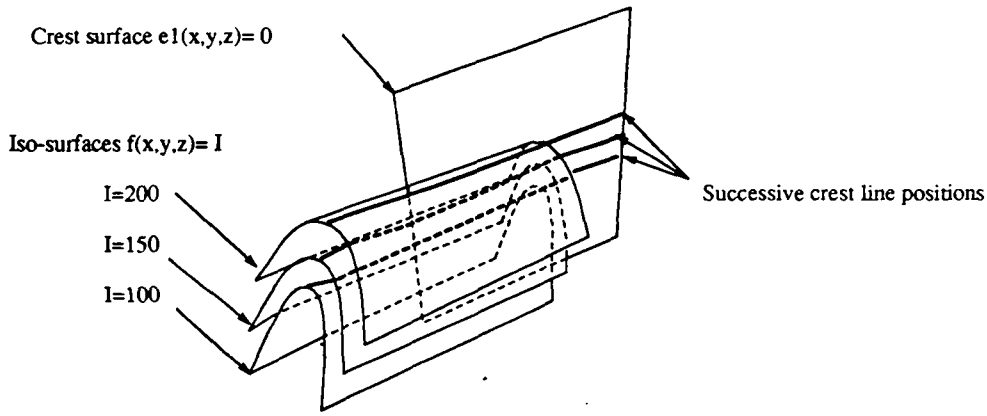


Figure 5: Crest surfaces in 3D images

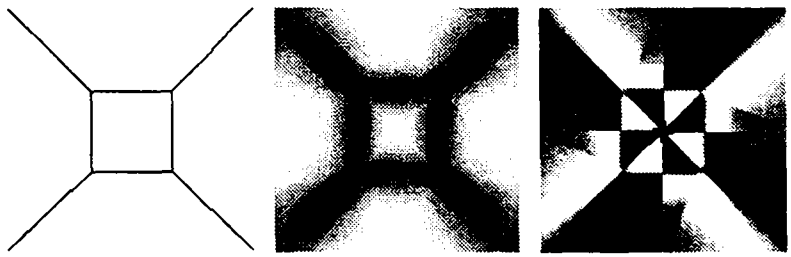


Figure 6: Crest lines, maximum curvature and extremality for $z = x^4 + y^4$

referred to as the "hairy ape" theorem, which states that a furry sphere cannot be combed without creating a cowlick, which means that there can't be a continuous field of vectors on a surface whose topology is spherical, without a pathological point where the associated vector is not defined (which is not true for a torus, for example, whose topology is different). Also, because the sign of the extremality function $e_1(x, y, z)$ is meaningless, the crest surface separates regions with $e_1 \geq 0$ from regions with $e_1 < 0$, but sometimes those regions can switch in the way of a chess board. This can be seen in the example of Figure 6, which corresponds to the function $z = x^4 + y^4$. If we threshold the zero-crossings according to the value of the principal curvature, the corresponding crest curves are not connected.

Instead, we propose to follow one crest line at a time, and to determine locally the orientation of the principal directions $(\vec{t_1}, \vec{t_2})$. More precisely, if the tangent of the crest line that we follow is $\vec{t}$ and the maximum curvature direction is $\vec{t_1}$, we orient $(\vec{t_1}, \vec{t_2})$ such as $\vec{t_2} \cdot \vec{t} \geq 0$, and also such that $\vec{t_1}$, $\vec{t_2}$, and the gradient direction $\vec{\nabla} f$ forms a direct trihedroni (see figure 2).

6 Experimental results in 3D

To adjust our implementation, we have first used synthetic data, corresponding to digital images of 3D functions for which the iso-surfaces constitute basic surfaces with known differential characteristics, such as a sphere, a torus, or an ellipsoid. Figure 7 corresponds to the computation of the extremality, the principal curvature directions and the extracted crest lines on the synthetic example of an ellipsoid (in fact, a 3D ellipsoidal function $f(x,y,z) = a^2x^2 + b^2y^2 + c^2z^2$). We can see in this image that the field of principal directions corresponds to the theoretical network of curvature lines of an ellipsoid, turning around the four umbilic points (on this view, only one of these umbilics can be seen). The sign of the extremality is displayed in green and blue, zero crossings correspond to crest lines (the white lines) or lines for which the curvature is minimal, (on top of the ellipsoid, linking the two umbilic points. The white line is the extracted crest line : an almost perfect ellipse. Those results are very encouraging, because the changes of curvature for the surface of the ellipsoid are very soft, and nevertheless, the extracted crest line is very stable. The addition of noise can be compensated to some extent by a larger σ constant when filtering with the gaussian function.

Figure 8 is the surface of the skin extracted from a 3D Magnetic Resonance image. This iso-surface is shown with four colors, corresponding to a classification of the surface into elliptic or hyperbolic regions (sign of the gaussian curvature), and rather convex or rather concave surface (sign of the maximum curvature). The four views correspond to different values of σ for the gaussian filters, which shows the effect of smoothing. The usefulness of those representations for 3D image interpretation has been shown, for example in [13] or [5], although they use a classical approach to the computation of the surface differential characteristics, where surface models are fitted to the 3D image to compute the differentials.

Figure 9 shows the iso-surface corresponding to the bones in a 3D CT scan of a head. The color corresponds to the value of the maximum curvature (with black points corresponding to high curvature points). We have performed the following experiment : we have extracted the crest lines from two 3D scans of the same patient, in two different positions (see figure 10). We have then applied a registration algorithm to the two lines sets (see [7], [15]), and resampled one of the lines set into the geometry of the other set. The superimposition of the registered lines sets is presented in figure 11, which shows the quality of both the crest lines extraction and the registration method. The quality is even more apparent when the mesh of 3D lines is rotating, or with a stereoscopic pair of images, because showing 2D projections is a poor way to visualize 3D lines. We have package the whole method into a single running software "epi_match", built with the EpidaureLib library of 3D image processing tools, which can be used to register automatically two 3D images, based on the extraction and matching of crest lines. This software is under current evaluation in the Harward Medical School, Boston, for a clinical study (with a potential of 1100 3D images to register!). The com-

putation time is about 30' on a DEC 5000 workstation, for two 256x256x128 images. The bottleneck of this software is the pre-filtering of the data with a smoothing filter using a large σ, for the two images (10' each). The extraction and matching part is comparatively faster (5' each).

At last, figure 12 shows the extraction, from an MR image, of the gyri of the brain, which are the crest lines with a positive maximal curvature. These results are promising, they show that the building of an atlas of anatomical crest lines can be foreseen, leading to the automatic interpretation and labeling of 3D images.

7 Conclusion

To conclude, we have described a new and very powerful method to compute the differential characteristics of the iso-intensity surfaces in 3D. It is new because it relies on the implicit definition of the surface, whereas the other methods make use of only their parametric representation. It is also very convenient because there is no need to fit and parametrize arbitrary surfaces to the data to compute the differentials of the surface. Our experiments on the automatic extraction of crests lines show the robustness of the computed differential characteristics, even if the computation of the differentials of the image are requested, up to order 3. At last, we have displayed several possible application of the computation of the principal curvatures, principal directions and extremality coefficient for the characterization of surfaces and the automatic registration of 3D images. We hope that in the future, our method will be extensively used to extract stable features for the automatic interpretation and labeling of 3D images.

Acknowledgment

We want to thank Olivier Monga, Serge Benayoun and Nicholas Ayache for stimulating discussions about crest lines, and also to Janes Wilhelms and Allen Van Gelder for discussions about iso-surfaces. We want to thank André Gueziec for his 3D line registration software. We want also to thank Digital Equipment Corp. who provides us with fast computers, GE-CGR of Buc, France, who provided the two scanner images of the head phantom. Thanks also to Maple, which helped us to perform the formal computations presented in this paper. Bob Hummel deserves special thanks for his careful review of an early version of this paper.

References

[1] Amir A. Amini and James S. Duncan. Differential geometry for characterizing 3d shape change. In *SPIE, Mathematical Methods in Medical Imaging*, volume 1768, pages 170–181, San Diego, USA, July 1992.

[2] Michael Brady, Jean Ponce, Alan Yuille, and Haruo Asada. Describing surfaces. *Computer Vision, Graphics, and Image Processing*, 32:1–28, 1985.

[3] Rachid Deriche. Recursively implementing the gaussian and its derivatives. In *Proc. of the 2nd Int. Conf. on Image Processing*, pages 263–267, Singapore, September 1992.

[4] Manfredo P. Do Carmo. *Differential Geometry of Curves and Surfaces.* Prentice Hall, 1976.

[5] Denis Friboulet, Isabelle E. Magnin, Andreas Pommert, and Michel Amiel. 3d curvature features of the left ventricle from ct volumic images. In *SPIE, Mathematical Methods in Medical Imaging*, volume 1768, pages 182–192, San Diego, USA, July 1992.

[6] Gerard Giraudon and Rachid Deriche. On corner and vertex detection. In *Conference on Computer Vision and Pattern Recognition*, Hawaii (USA), June 1991.

[7] A. Guéziec and N. Ayache. Smoothing and matching of 3D-space curves. In *Proceedings of the Second European Conference on Computer Vision 1992*, Santa Margherita Ligure, Italy, May 1992.

[8] Les Kitchen and Azriel Rosenfield. Gray-level corner detection. *Pattern Recognition Letters*, 1:95–102, 1982.

[9] Jan J. Koenderink. *Solid shape.* The MIT Press, 1990.

[10] Oivier Monga, Serge Benayoun, and Olivier D. Faugeras. Using partial derivatives of 3d images to extract typical surface features. In *Proceedings CVPR '92, Urbana Champaign, Illinois.* IEEE, July 1992. also an INRIA Research Report (1599).

[11] Olivier Monga, Nicholas Ayache, and Peter Sander. From voxel to curvature. In *Computer Vision and Pattern Recognition*, pages 644–649. IEEE, June 1991.

[12] Alison J. Noble. Finding corners. *Image and Vision Computing*, 6:121–128, 1988.

[13] Peter T. Sander and Steven W. Zucker. Singularities of principal direction fields from 3-d images. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 14(3):309–317, March 1992.

[14] J-P. Thirion and A. Gourdon. The 3d marching lines algorithm and its application to crest lines extraction. *rapport de recherche INRIA*, (1672), May 1992.

[15] J-P Thirion, O. Monga, Benayoun S., Gueziec A., and Ayache N. Automatic registration of 3d images using surface curvature. In *IEEE Int. Symp. on Optical Applied Science and Engineering*, San-Diego, July 1992.

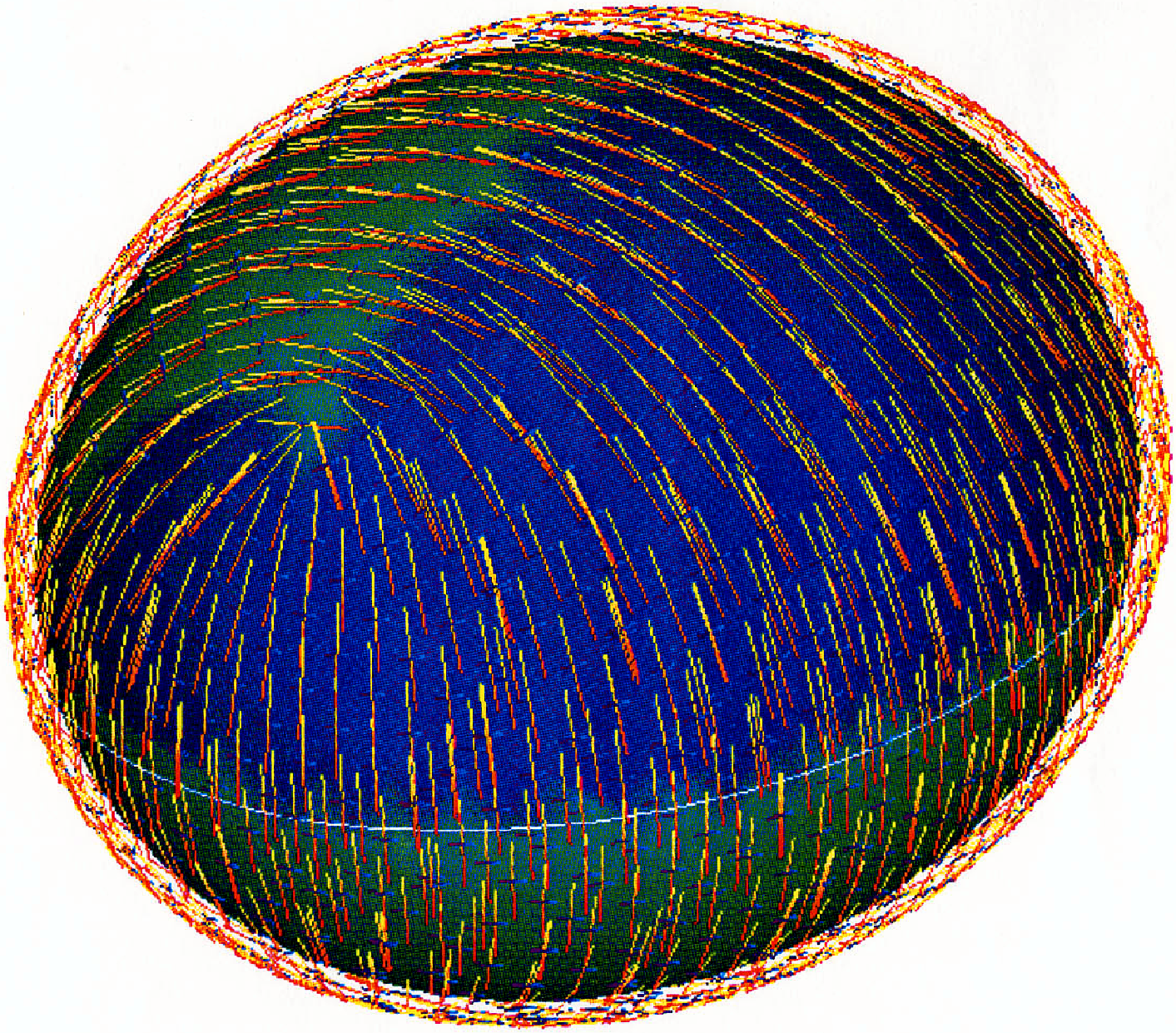


Figure 7: The iso-surface is extracted from the 3D synthetic image of a 3D ellipsoidal function. This iso-surface is colored with the extremality function values. The white line is the extracted crest line (an ellipse). The vector field corresponds to the principal directions computed with our method.

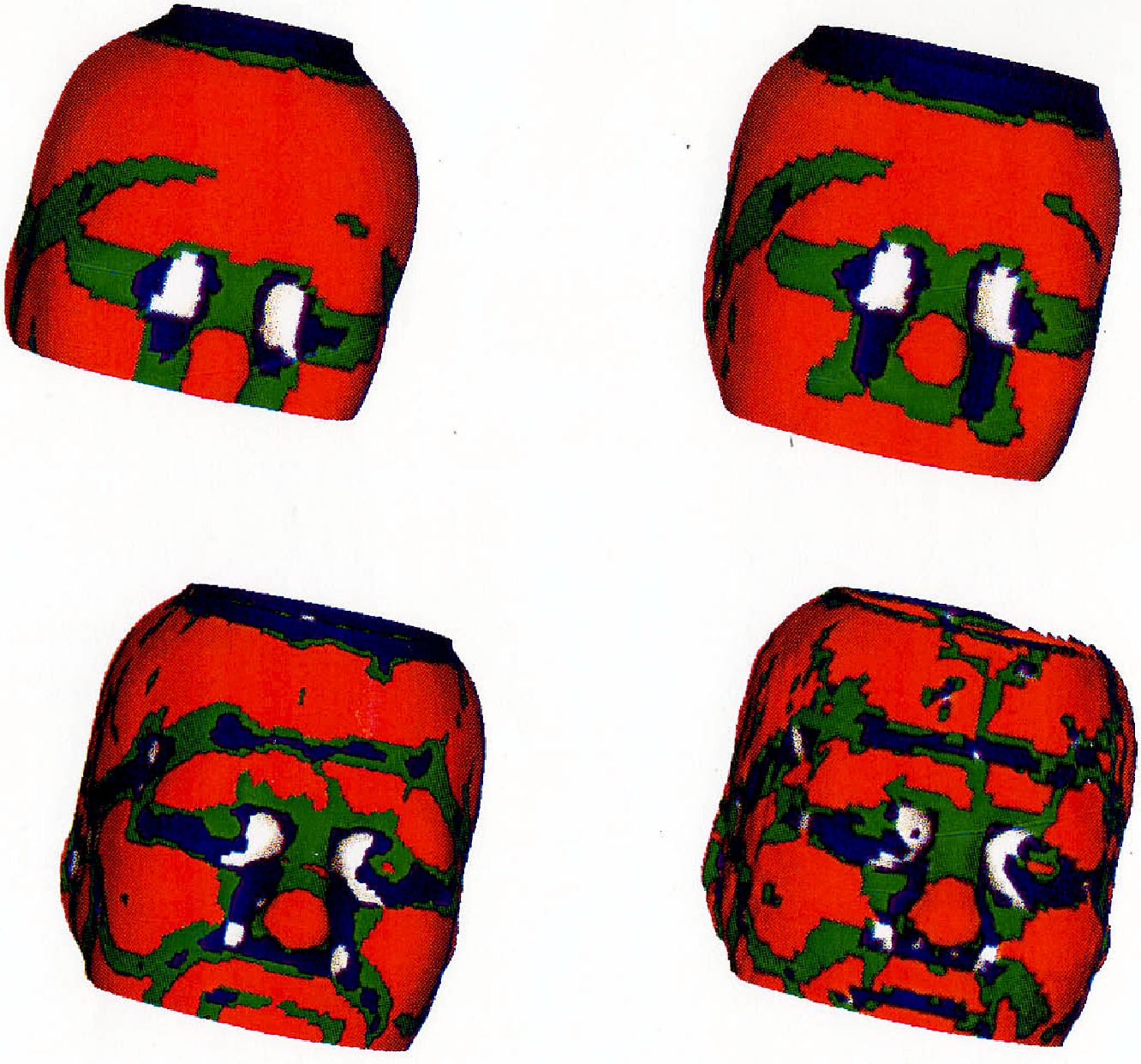


Figure 8: Computation of characteristic regions on the iso-surface of the skin, in a 3D MR image, and for different smoothing factors. Red: convex elliptic, white: concave elliptic, Green: convex hyperbolic, Blue: concave hyperbolic.

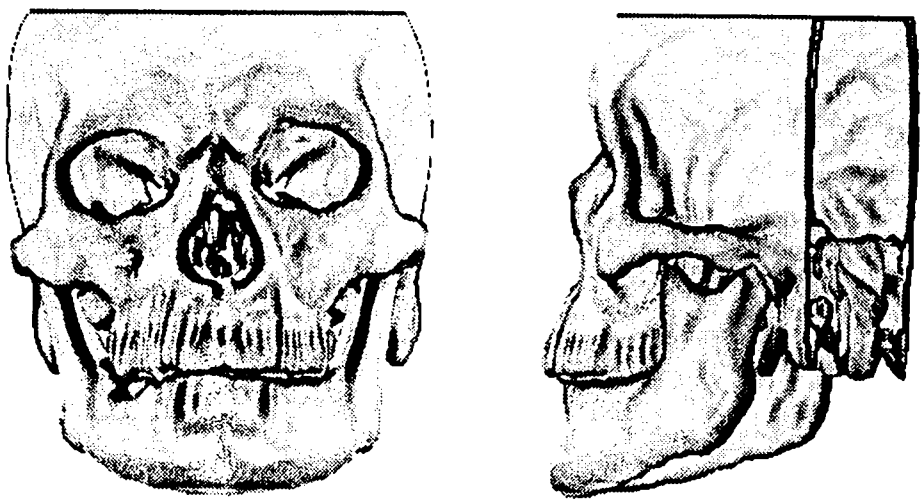


Figure 9: Maximal curvature on the iso-surface of the skull, for a 3D XRay image. Black corresponds to high values of the maximal curvature.

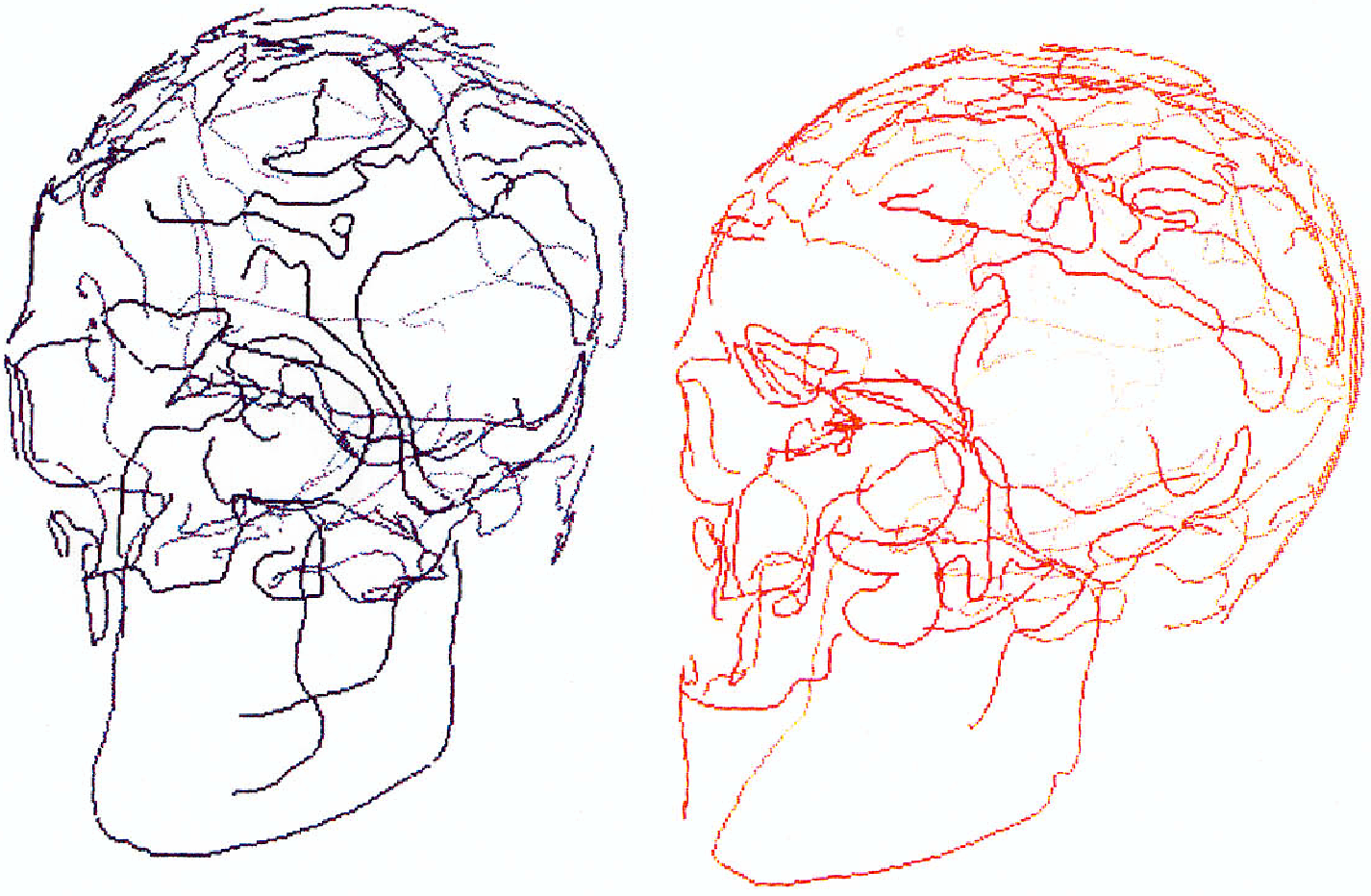


Figure 10: Filtered crest lines for two 3D data sets

Figure 11: 3D registration of the two crest line sets (red and blue)

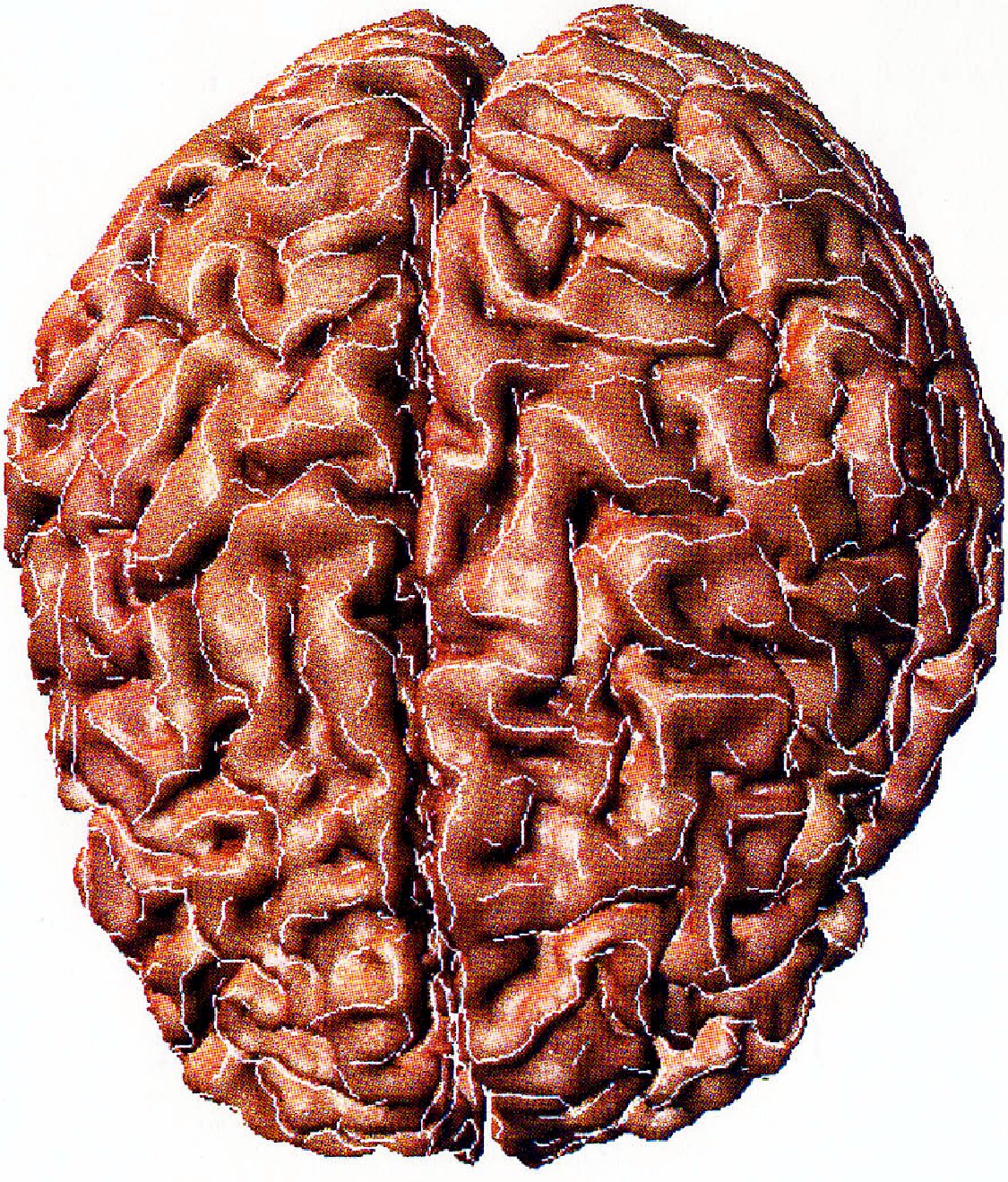


Figure 12: This is another example of crest lines extraction on a 3D Magnetic Resonance image. The crests correspond to the circumvolutions (gyri and sulci) of the brain. Coupled with an anatomical atlas of the crest lines, a possible application is for the automatic labeling of 3D MRI images.

The 3D Marching Lines Algorithm

Jean-Philippe THIRION and Alexis GOURDON
INRIA, Equipe Epidaure
B.P. 93, 2004 route des Lucioles
06902 SOPHIA ANTIPOLIS Cedex, FRANCE
Email: jean-philippe.thirion@inria.fr
Tel: (33) 93 65 76 62
Fax: (33) 93 65 79 69

Abstract

This paper presents a powerful and general purpose tool designed to extract characteristic lines from 3D images. One major contribution of this paper is the proof of the good topological properties of the reconstructed surfaces and 3D curves. Our algorithm, called the *Marching Lines*, is inspired from iso-intensity techniques, such as the Marching Cubes algorithm. The Marching Lines algorithm extracts, with sub-pixel accuracy, the 3D lines corresponding to the intersection of two iso-surfaces in a 3D grid. In fact, it is more general because it can be used to trace any implicit 3D curve, defined as the intersection of two implicit surfaces $f(x,y,z) = I$ and $g(x,y,z) = J$. We prove that our algorithm produces 3D lines which are continuous, not self-intersecting and closed (except of course at the boundary of the 3D grid). At last, we show several applications of the 3D Marching Lines to the automatic extraction of characteristic 3D curves such as the parabolic lines or the crest lines. We present also experimental results on synthetic and real 3D images, and an application to the registration of two 3D images, which proves the high quality of the extracted feature lines.

Key words: 3D Image, Iso surface, Crest Line, Topology, Differential Geometry.

1 Introduction

Image interpretation generally consists in extracting symbolic information from n-dimensional images. The goal is to get a smaller and more manageable representation of the information. We present in this paper a new method to extract characteristic lines from 3D images. This subject is relatively new because 3D isotropic images have emerged only recently, and people were more concerned with the extraction of surfaces or of characteristic points than with 1D varieties of points (lines). However, depending on the application, and on the computation time requirement, any kind of symbolic information can be useful: lines as much as points or surfaces.

The new method that we present here, which we call the Marching Lines algorithm, is an efficient tool to extract automatically characteristic lines out of 3D images (first described in [15]). In fact, in a more general application, it can be used to compute 3D curves defined by two implicit equations $f(x, y, z) = I$ and $g(x, y, z) = J$. It is an extension of iso-itensity surfaces, which are used to extract implicit surfaces defined with only one equation $f(x, y, z) = I$.

Conversely to many other works, we not only give the description of the algorithm, but also a proof that the reconstructed surfaces and lines are topologically correct.

The output of the Marching Lines is a list of 3D oriented curves, each curve being defined with a linked list of 3D points, the order of the points reflecting the orientation of the curve. The input depends on the kind of application. In its more general application, it can be two continuous functions, which give, for each point (x, y, z) of the 3D space, two real values : $f(x, y, z)$ and $g(x, y, z)$. We show that the reconstructed curves are not self intersecting and complete (i.e. closed curves), and explain the signification of their orientation.

The first section gives the definition of an iso-surface, and describes its topological properties. We describe then an algorithm to extract iso-contours, first in 2D, and then in 3D. The major contribution of this section is the proof of the good topological properties of the extracted iso-surface, such as the completeness. Then we describe the Marching Lines algorithm, regardless the source of the two 3D functions f and g. We explain how the algorithm can produce *oriented* 3D curves, with also "good" topological properties. At last, we present experimental results of the automatic extraction of lines,

for both synthetic and real 3D images, such as the parabolic curves, the iso-gaussian curves, and the crest lines.

2 About the topology of iso-surfaces

This section is a philosophical dissertation about the definition of iso-surfaces. It gives orientation conventions which are indispensable to the understanding of the next sections (there are 4 independant orientation conventions in this paper). We describe the topological properties of iso-surfaces, and show that there is no qualitative difference between the topological properties of the continuous world and the discrete versions of it. Particularly, we explain that the arbitrary choice which has to be made for the topology of the discrete world, between the "object" and the "background" (for example between the 4 and 8 connexity for the 2D digital images), has also to be made for the continuous world.

A surface can be defined either in a parametric way, as a local map of $\mathcal{R}^2$, or in an implicit way as the set of points that verify an implicit equation $f(x, y, z) = I$, where I is a constant, these are the "ideal" visions of surfaces. But not all the continuous function f of $\mathcal{R}^3$ define a surface, $f(x, y, z) = I$ can also defines regions of homogeneous intensity I.

We present now another, less conventional but more intuitive way to define a surface. A surface can be defined as the iso-intensity surface S_1 (in short iso-surface) of a continuous function f of $\mathcal{R}^3$, in the following way :

convention 1 (The object/background convention) *The iso-surface is the interface between the regions of the space where $f \geq I$ and the regions where $f < I$, I being the iso-value constant.*

We could have chosen also the interface S_2 between regions where $f > I$ and regions where $f \leq I$. This is an *arbitrary* (and unavoidable) choice to make if we want it to be a surface, and not sometimes a volume. The surfaces S_1 and S_2 are exactly the same, except for the boundaries of the constant intensity regions $f = I$ (see figure 1).

Let's consider now the properties of those iso-surfaces. S_1 is a complete and oriented surface (S_2 also). Complete means that S_1 has no holes, nor edges (see for example [7], p. 343). We can orient S_1 from the regions $f \geq I$, the inside, toward the regions $f < I$, the outside. Another property of S_1 is

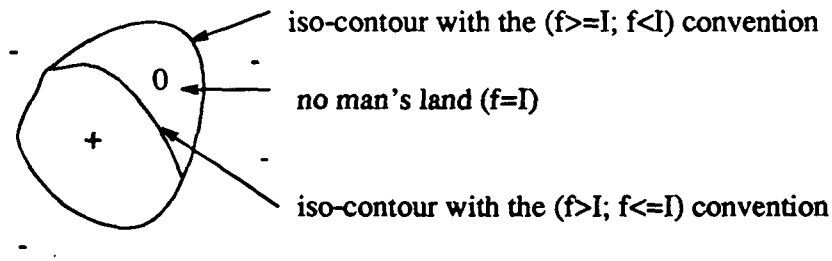


Figure 1: Definition of iso-contours in 2D

that it doesn't intersect itself. A Klein bottle, for example, for which there is no inside nor outside, and which is self intersecting, can't be defined as the iso-surface of a continuous function.

In the rest of this paper, we will take the S_1 definition of iso-surfaces. The words : "object", "white", "+" or "inside" are synonyms for the regions $f \geq I$, whereas "background","black","−" or "outside" are synonyms for the regions where $f < I$. To justify this, the opposition $+-$ refers to the sign of $f - I$, 0 being considered arbitrarily as a positive number. The justification of "object" for $f \geq I$ is much more interesting : we have prefered to associate an "object" to a topologically closed set. We personally can't imagine what an "open" set is either as the complementary of a closed set, in other words, as the background of an object. This leads to the definition of the inside and the outside, the inside being the inside of the object (few people speak of the inside of the background). In turns, it defines the inside and outside face of the iso-surface S_1. To sumarize this :

$$f \geq I = \text{Closed set} = \text{Object} = \text{Inside}$$
$$f < I = \text{Opened set} = \text{Background} = \text{Outside}$$

As we can see, the arbitrary choice of the S_1 definition of iso-surfaces is strictly the equivalent to choosing a sigh for 0, choosing what is the object versus the background, choosing the eight connectivity of the white pixels for digital images, or choosing what is the inside versus the outside. We will refer later to this arbitrary choice as the "object/background convention", which is not particular to digital topology as many people think. To sumarize the equivalent arbitrary choices :

sign of 0 $\Leftrightarrow$ closed/open $\Leftrightarrow$ object/backgrownd $\Leftrightarrow$
8/4 connectivity $\Leftrightarrow$ inside/outside

Let's take an example. In the "real" world (the one composed with atoms), the notion of surface vanishes when it is considered at a very small scale. But at a macroscopic scale, what is commonly called a surface is in fact an interface. We take for granted that a sheet of paper is an object, whereas the air is the background. With our definition, the surface of the sheet of paper is still complete (i.e. without edges) : if you look very close to it, you will see that the surface rounds continuously around its edge and prolongates on the other "face". In fact, a sheet of paper is nothing else than a very thin parallelepiped. The surface of the sheet is oriented from the paper matter (the inside) toward the air (the outside). In that way, such a complex object than a physical moebius strip (the object) has still an oriented boundary surface; we can see only the outside face of it.

That an iso-surface is always complete does not limit the topology of the enclosed object. In fact the enclosed object can be a torus, or a volume of any genus (i.e. number of ankles), and even an object with interleaved ankles, which means that the iso-surface definition is powerful enough to represent any real object boundary (except when we look to close to the atoms).

3 The extraction of iso-surfaces

We describe here previous works about iso-surfaces extraction, and present a new method which ensure that the topological properties of iso-surfaces are preserved.

A 3D images, such as Magnetic Resonance Images (MRI), or the compilation of Xray scanner images, is a discrete and bounded space. We can go from the discrete space to the continuous space with the convolution of the image with a continuous function, or with an interpolation function. We find back a continuous intensity function $f(x, y, z)$ which is a good approximation of the true one, up to the reduction of its power spectrum.

The view point of discrete topology (see for example [5], [8], [4], [13]) is to consider the digital image with an interpolation of order 0, that is, replacing digital samples, the "voxels", with cubes of constant intensity, and sometimes, with only binary values. The extracted iso-surface is then the

collection of the voxels which are traversed by the iso-surface of the true object. This can be used efficiently for visualization, especially for high resolution images, but this representation is not well suited in many other cases.

Other people have successfully proposed to extract iso-surfaces with higher order interpolation functions (see [20],[10], [2], [18], [9]). A good survey of those methods can be found in [19] or in [6]. A classical example is the Marching Cubes algorithm, as defined by Lorensen and Cline in [10], which corresponds, to some extend, to the use of the tri-linear interpolation. To summarize the Marching Cubes, as defined in [10], the eight vertices of each voxel can be labeled with '$-$' or '$+$', according to the value of the vertex with respect to the selected iso-value (the *object/background* convention is not discussed). This defines a 8-bit code (that is, 256 cases) for each voxel. A fixed triangulation of the iso-surface in this voxel is then proposed. The position of each triangle is adjusted such that the triangle vertices corresponds to the linear interpolation of the iso-surface along the edges of the voxel. In [10], the 256 possible cases are reduced to 15 cases of triangulation with the use of symmetry. However, the method presented in [10] does not ensure that the reconstructed surfaces is complete : in fact there are some holes in it (noticed first in [1]).

This is the major problem of the proposed surface reconstruction algorithms. Some are topologically correct (and other are not), but the authors fail to prove why. So far, the only complete proof of a "topologically correct" reconstruction algorithm that we found is the work of Kalvin in [6]. The method that is presented, however, is not local, but propagates constraints during the reconstruction in order to ensure good topological properties. Another interesting attempt to rationalize iso-surface extraction is the work of Wilhelms and Van Gelder in [19]. In the present paper, we show that even a *local* algorithm, that is, when the computation in one voxel is independant from the computation in the other voxels, can be used to ensure good *global* topological properties. We have identified independent orientation conventions for this (one being the object/background convention) which can be useful to prove other algorithm of surface or line reconstruction.

3.1 The 2D case

We can define a curve C_1 on an oriented surface S_1, as being the interface on S_1 of a given continuous function $f(x, y, z)$. C_1 separates the regions of S_1 where $f \geq I$, the inside, from the regions of S_1 where $f < I$ (the *object/background* convention). We introduce here another convention, the well known *left handed* orientation of the curves on oriented surfaces :

convention 2 (The left handed orientation convention) *An observer (Ampere Goodman is its name), following the curve C_1 on the surface S_1, having its feet in the inside of S_1 and its head in the outside of S_1, will always have the inside of C_1 at is left.*

We describe now the restriction of our method to two-dimensional images. An infinite plane is a complete surface, we suppose that we look at this infinite plane from the outside. A 2D image is the sampling of a continuous function f on a regular grid. We want as output a list of oriented curves, reflecting the iso-contours of f, corresponding to an iso-value constant I. The iso-contours are defined without ambiguity with : the image, a given interpolation function, an iso-value I and the use of the *object/background* convention. We choose here the bi-linear interpolation function, but other functions can be chosen.

Let's consider one of the cell of the grid (a square). Each vertex can be labeled with "+" or "−", according to the *object/background* convention, which leads to 16 cases for a square, which can be reduced to seven with geometric symetry considerations (Figure 2, see also [20]). If at least one of the vertex label differs from the others, the iso-lines defined with the linear interpolation correspond to an hyperbola passing through the cell. The two branches of the hyperbola are separated with asymptotic lines which are *isothetic* (i.e. parallel to the major axes). We can approximate the branches of the hyperbola with segments, intersecting the hyperbola at the sides of the square. Our algorithm corresponds to the linear interpolation only "to some extent" because of this approximation (see figure 2).

Each of the four segments which compose the sides of the cell can be intersected in at most one point by the hyperbola, except for degenerated cases. To have a more practical implementation, we get ride of those degenerated cases once for all by adding a small positive constant value "ε" to

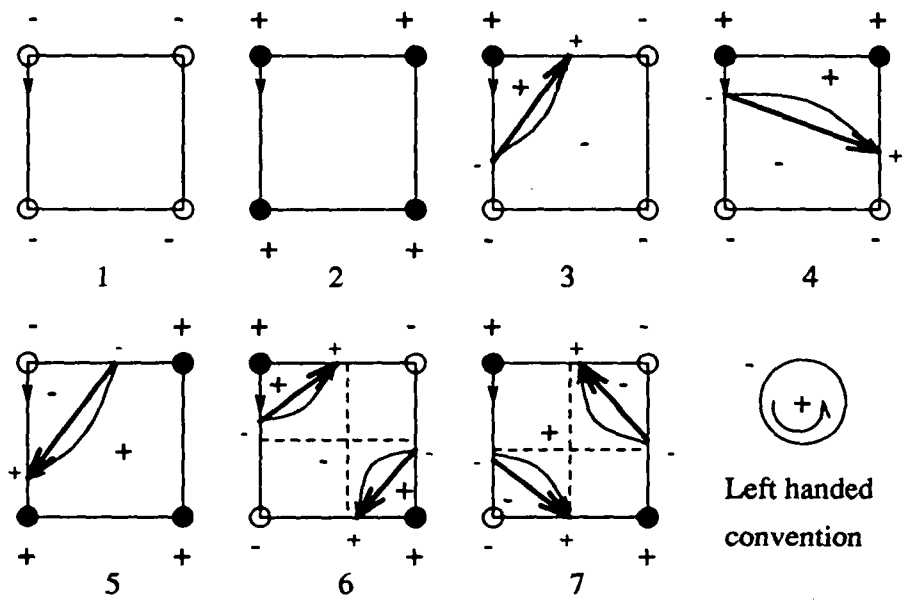


Figure 2: Cases of segmentation of the cell in 2D

each vertex where $g = I$; this doesn't perturbate the *object/background* convention. Other solutions can be used, such as shifting the intersection point along the side of the cell, but to preserve good topological properties, this has to be made with care, for example, by shifting always in the directions of increasing coordinates.

Furthermore, the addition of ε ensures that the intersection with the cell sides lays always strictly inside those segments. We define a procedure, $(Q, L) = INTER(v1, v2, J)$, where $v1$ and $v2$ are the intensity value at the vertices (P_1, P_2) of a unit cell edge, and I is the iso-value, and which gives back either nothing if the labels of P_1 and P_2 are the same, or the interpolated position Q of the intersection of the iso-contour with $]P_1, P_2[$ (strict because of the ε shift). L is the label of P_2. We apply this procedure to the four sides of the cell, oriented according to the *left handed* orientation convention.

lemma 1 *The application of* INTER *to the four vertices of a cell is an ordered list of n labeled intersection points* (Q_i, L_i)*, where n is* 0,2 *or* 4*. Furtermore, their labels are alternatively* $+$ *and* $-$*.*

proof: see figure 2.

We define now the procedure *BUILDSEGMENTS*, which takes as input all the labeled points produced by the application of *INTER* to the sides of the cell. For $n = 2$, we constitute one segment, oriented from $-$ to $+$. For $n = 4$, we form two segments oriented from points labeled $-$ to points labeled $+$. We can notice that there is an ambiguity in that case, which is reflected by Case 6 and Case 7 of Figure 2. This choice has no matter in 2D.

lemma 2 *The result of* BUILDSEGMENTS *is* 0,1 *or* 2 *oriented segments, coherent with the* left handed *convention in that the vertices of the cell lying in the left region defined with the oriented segments have a label* $+$*.*

proof: see figure 2).

theorem 3 *:*

the set of oriented segments provided by the application of BUILDSEGMENTS *to the whole set of cells in the image defines oriented, not self-intersecting, and complete curves, except at the boundary of the image.*

Proof: Each side of a cell which is not at the edge of the image is shared with exactly one other cell. Each cell boundary is oriented with respect to the *left handed* orientation convention. The orientation of this side is opposite for the two cells. The procedure *INTER* is deterministic and produce the same intersection point Q, if any. The label L of this point is opposit for each of the cell, therefore, exactly one oriented segment arrives at this point Q in the cell where its label L is $+$, and exactly one segment starts from Q in the cell where its label L is $-$. Thus, for every point Q produced by this algorithm, which doesn't lay on the boundary of the image, there is strictly on incoming and on outgoing oriented segment. The not-self intersecting properties can be verified on figure 2.

One word about the ambiguity of case 6 and 7, which is of no importance for the 2D case, but which has to be raised for the 3D case. One way is to compute the average value $\overline{v}$ of the four vertices of the cell, and to compare it with the iso-value threshold I. The case $\overline{v} = I$ has to be considered with special care (see figure 3). In that case, we can either use a special case of segmentation (Case 2), or arbitrarily choose one of the solutions (Case 1 or Case 3). To be coherent with the *object/background* convention, if $\overline{v} \geq I$ we choose Case 7 else if $\overline{v} < I$ we choose Case 6. In fact, this solution does not lead to the same connectivity as the one defined for the continuous image obtained with the bi-linear interpolation of the discrete image. The exact solution is to compute the value of this bi-linear interpolation at the crossing point of the two asymptotes of the hyperbola, which corresponds, with the notation of Figure 3, to compare the value of $\hat{v}$ (see equation 1) with the iso-value I and the *object/background* convention. In the rest of the paper, we will suppose that this convention is used to raise the ambiguity in *BUILDSEGMENTS*.

$$\hat{v} = (s_1 s_4 - s_2 s_3)/(s_1 + s_4 - s_2 - s_3) \tag{1}$$

What is remarkable is that *BUILDSEGMENTS* can be applyed independently to each cell of the grid, and nevertheless, the reconstructed curves have all the good requiered topological properties of the iso-contours. Another way to use *BUILDSEGMENTS* is to apply it to one cell taken at random, and to follow the iso-contour if there is one coming through that cell, "marching" from this cell to the next one. This is why we call this algorithm, the

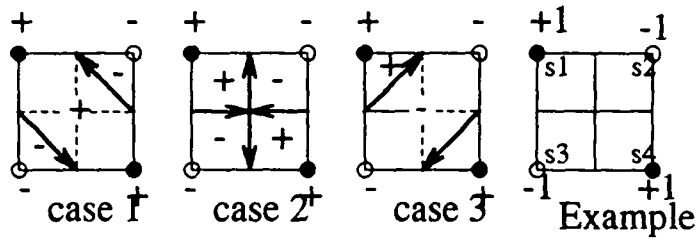


Figure 3: Iso-boundaries: singular situation

"2D Marching Lines". Taking several seeds at random permits to extract the main iso-contours of the image with much less computation than the extensive application of *BUILDSEGMENTS* to the entire image (for example, we applied this in [14]), which constitutes the randomized application of the Marching Lines.

3.2 The extraction of Iso-surfaces

We describe now an algorithm which extracts an iso-surface from a 3D digital image, for which we demonstrate that the topological properties are preserved (mainly the completeness).

A 3D image is a 3D regular grid composed with cubic cells, with the sampling values of a continuous function f at each vertex. The tri-linear interpolation defines cubic surfaces whose intersection with isothetic planes are hyperboles. In one cubic cell, the cubic surface can be composed with several disconnected patches. We call each of these patches an *iso-patch.* The iso-surface is defined without ambiguity with the 3D image, the interpolation function, a given iso-value I, and the application of the *object/background* convention. Our method has good topological properties because it reflects the topological propertie of this iso-surface.

Our method consists in approximating the iso-patches by 3D polygons, each of these 3D polygons being a linked list of 3D points constituting non-necessary plane polygons (see figure 4).

The first step of our method is to consider each face of the cubic cell, and to orient it toward the outside of the cell. Each face is a square, and we orient the segments corresponding to the sides of this square with the *left handed*

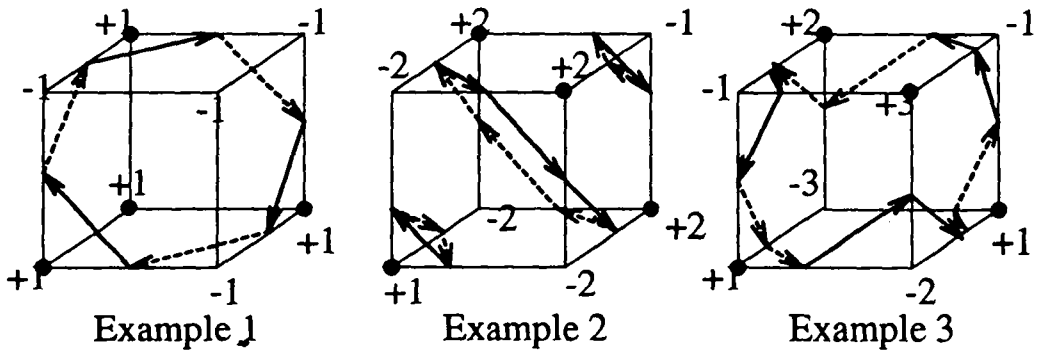


Figure 4: Examples of oriented 3D polygons in a voxel

orientation convention. We can therefore apply the *BUILDSEGMENTS* procedure, which gives us 0, 1 or 2 oriented segments, whose intersections with the sides of the square are exactly the intersection of the iso-surface with this face of the cube cell. We suppose also that the *object/background* convention has been applied for the $\hat{v}$ value of equation 1.

The new procedure *BUILDCYCLES* is the application of *BUILDSEGMENTS* to the six faces of the cubic cell, oriented toward the outside of the cell, it takes all the oriented segments which are produced, and constitute oriented 3D polygons with them.

lemma 4 *the application of* BUILDCYCLES *to a cubic cell produce complete (i.e. closed) oriented curves (the 3D polygons). The maximum numbers of segments produced is 12, and the maximum number of 3D polygons is 4.*

Proof: The proof that the oriented segments consitute closed oriented curves around the cubic cell is similar to the proof of theorem 3; each side of the cube is considered twice, because it is shared by exactly two adjacent faces. Furthermore, the orientation being opposite, the associated label is also opposite, thus there is exactly one incoming and one outgoing oriented segment for each generated point. The bound 12 is because there is at most 2 segment per faces, and the bound 4 because there are no degenerated 3D polygons (due to the ε shift), therefore they are composed with at least 3 segments.

Those 3D polygons do not intersect on the faces of the cube, because *BUILDSEGMENTS* produce no crossings. Examples of cycles are presented in Figure 4. Cases 2 and 3 are not referenced in the original Marching Cubes algorithm description, which shows that our algorithm leads to different solutions. Furthermore, those two examples have the same associated 8-bit codes and nevertheless lead to different triangulation cases. Finally, we can see in example 3 that the case with 12 vertices can occur. We can also demonstrate that 3D polygons with 10 or 11 vertices are impossible.

theorem 5 *The application of* BUILDCYCLES *the the entiere set of grid cells is a set of oriented 3D polygons which constitute oriented, not self-intersecting and complete surfaces, except at the grid boundary.*

Proof: The proof is once again similar to theorem 3. Each face of a cubic cell, which is not at the boundary of the grid, is shared by exactly two cubic cells, and their orientations are opposite. The application of *BUILDSEGMENTS* is deterministic (due to the use of $\hat{v}$, and the *object/background* convention). The produced segments are therefore exactly the same for the two applications of *BUILDSEGMENTS* to the face, except for their orientation, which is exactly opposite. Each segment in the face is therefore shared by exactly two 3D polygons, one in each cell, and its orientation is opposite for the two 3D polygons, which proves that those two 3D polygons have a compatible orientation. Because there is no segment belonging to only one 3D polygon (except at the boundary of the grid) the reconstructed surface can't have a "holes" (holes always have edges), therefore, the reconstructed surface is both oriented and complete.

It exactly corresponds to the iso-surface defined with the tri-linear interpolation at the intersection with the sides of the grid, which proves that the thiner the grid is, the closer our solution is from the exact iso-surface.

At last, we need a way to transform the 3D polygon into a manageable representation of a surface. Many solutions hold, provided that the reconstructed surface doesn't intersect another iso-patch in a grid cell. One solution is to take the minimal area surface of each 3D polygon. Another way is to triangulate them. When the number of vertices is greater than 3 the surface can be triangulated in many different ways, as can be seen in Figure 5. We suggest triangulating the polygon with respect to its barycenter (Choice 2).

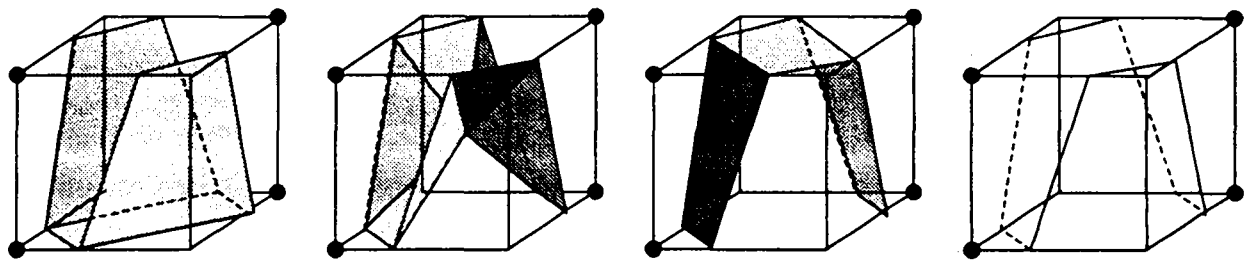


Figure 5: Different choices for the triangulation

In that way, the reconstructed surface is oriented, complete (except at image boundary), and doesn't intersect itself, which are all the good topological properties of iso-surfaces.

Another variation of this method is to use the decomposition into tetrahedra of the voxels as proposed by Payne and Toga in [12] in the case of iso-surface computation : each grid cell can be decomposed into 5 tetrahedra. We have at most one iso-surface within each tetrahedron, which can be approximated with one triangle. Using properly the *object/background* and the *left handed* convention we can prove also that this algorithm produce iso-surfaces with the requiered topological properties. Some criticism appeared concerning the isotropy of this solution, because the tetrahedra decomposition is not the same from one voxel to the other one (see for example [6]). This problem is reduced when the resolution gets higher. It is however not clear to state which of the two proposed algorithms (cubes or tetrahedra) is the easiest to implement, because managing the orientation of tetrahedra is not straitforward, and the total number of triangles produced by the tetrahedra method is larger.

4 The 3D Marching Lines algorithm

Now that we are ensured that the reconstructed iso-surfaces have good topological properties, let's consider how to compute the intersection of two iso-surfaces S_1 and T_1. We call the corresponding lines *bi-iso-lines*, ("bi" in

order not to mistake them with iso-contours in 2D, and refering to the pair of surfaces). The bi-iso-lines are almost entirely defined by two 3D digital images, two corresponding interpolation functions, two iso-values I and J, and the use of the *object/background* convention : the intersection of the two sets of surfaces computed with the algorithm defined in the previous section is one solution to find the bi-iso-lines. This is true, except for the regions where the two surfaces are tangent, for which a new orientation convention is needed, as we will see in this section.

For 3D images, the number of grid cells can be very high, which means that a simpler algorithm to compute the bi-iso-lines segments will be well-comed. This is the case for the algorithm that we present now. Furthermore, we prove that the reconstructed bi-iso-lines are oriented, not self-intersecting and complete (which means closed curves), except of course at the 3D image boundaries.

4.1 Orientation considerations

Besides the *object/background* and the *left handed* orientation convention, we need another convention regarding to the intersection of two oriented iso-surfaces S_1 and T_1. Let (f, I) be the continuous function and iso-value defining S_1, and (g, J) the definition of T_1. If there are no degenerated cases (tangency of the surfaces), the intersection of S_1 and T_1 are 3D curves. If we look at the values of function g on the surface S_1, they define iso-intensity contours which separates the regions of S_1 where $g \geq J$ from the regions where $g < J$ (the *object/background* convention). These 3D curves can be oriented with respect to the *left handed* convention. We can do the same when looking at the values of f on the surface T_1 : the oriented curves are exactly the same, except that the orientation is opposite (see figure 6). Except also for the boundary of tangency regions, as we can see in figure 6 (a phenomenon somewhat similar to the constant intensity regions of iso-surfaces). With anyone of these two definitions of bi-iso-lines, there is no degenerated case : even if the two surfaces are tangent, the bi-iso-lines are still lines, which is a remarkable property of those two definitions. The following orientation convention consists in arbitrarily choosing one of those two definitions :

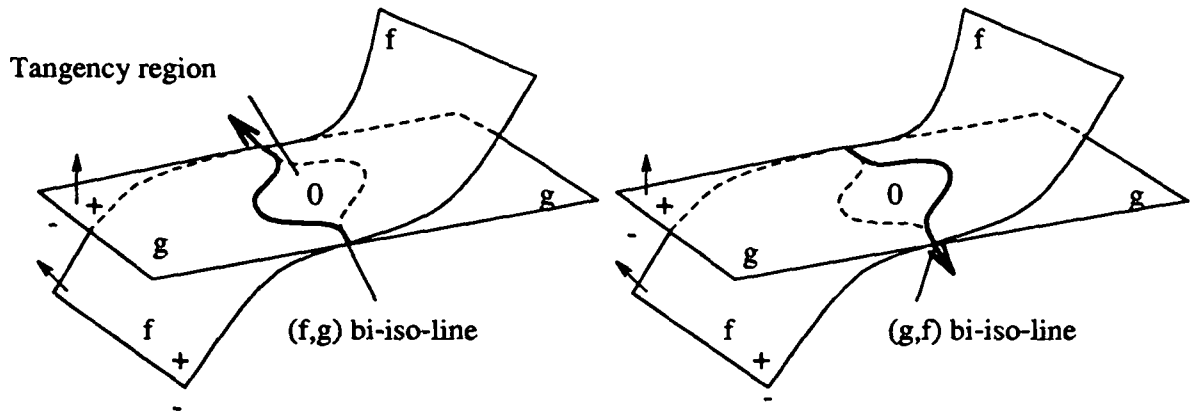


Figure 6: Definition and orientation of the bi-iso-lines

convention 3 (*bi-iso-line* **orientation convention**) *Let* (S_1, T_1) *be an ordered pair of iso-surfaces defined with respectively* (f, I) *and* (g, J). *The bi-iso-lines associated with* (S_1, T_1) *are the oriented iso-contours of the function* g *on the iso-surface* S_1.

4.2 Description of the algorithm

The 3D space is once again sampled with a regular 3D grid, but this time, two ordered values f, g are defined for each vertices. Those two values may come from two different images, but can also be computed from the same image with two different functions, as we will see later.

We describe now one possible implementation of the algorithm, and also the proof of the good topological properties of the reconstructed 3D curves. Let us consider one cubic cell of the grid. The application of *BUILDCYCLES* to the cell, corresponding to the values f, produce a given number of 3D polygons. By construction, the vertices of those 3D polygons lay strictly on the edges of the cubes. For each of those intersection points, we compute by linear interpolation along the edges (but once again, other interpolation functions can be used) the values of g at those points, as can be seen in figure 7. This is the application of the *bi-iso-line* convention. We have then a value

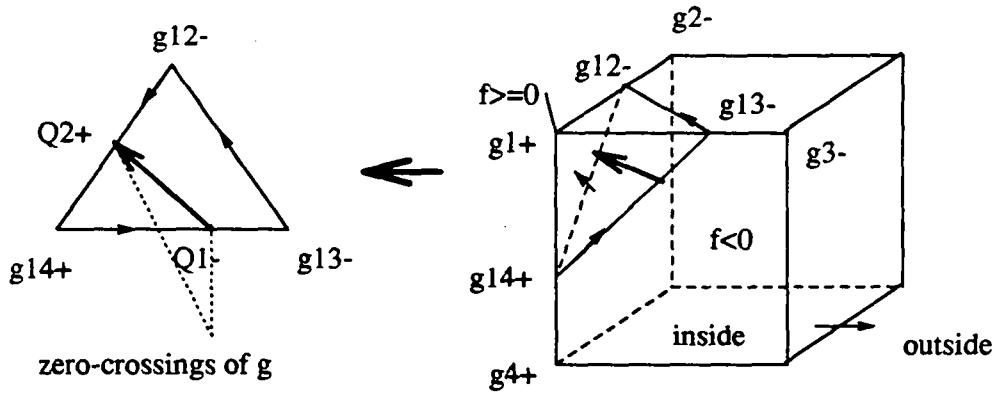


Figure 7: The computation of bi-iso-lines segments

asociated to each point of the 3D polygons, and we can label those points with respect to the iso-values J and the *object/background* convention.

We call *CYCLESTOPOINTS* a new procedure which takes such a 3D polygon, and which produce an ordered list of labeled points in the following way : recall that the 3D polygon defines an oriented surface by the *left handed* convention. It is composed with oriented segments, valued with the function g. We can apply the *INTER* procedure to each of these oriented segments, which gives an ordered list of labeled interpolated 3D point $\{(Q_i, L_i)|i \in [0..n]\}$, lying strictly in the inside of the oriented segments of the 3D polygon. Figure 7 is an example of this for a 3D polygons composed with only 3 points, in that case, two labeled points are produced : $\{(Q_1, -), (Q_2, +)\}$.

lemma 6 *If* $\{(Q_i, L_i)|i \in [0..n]\}$ *is the ordered list of 3D points produced by* CYCLESTOPOINTS, *then* n *is pair, and the labels* $\{L_i\}$ *are alternatively* $+$ *and* $-$.

Proof: based on the theorem of the intermediate values, a given number of times.

Furthermore, the maximum number of 3D points, n, is limited to the number of points constituting the 3D polygon, and therefore doesn't exceeds 12. A new procedure *CYCLETOSEGS* takes a 3D polygon, and gives back

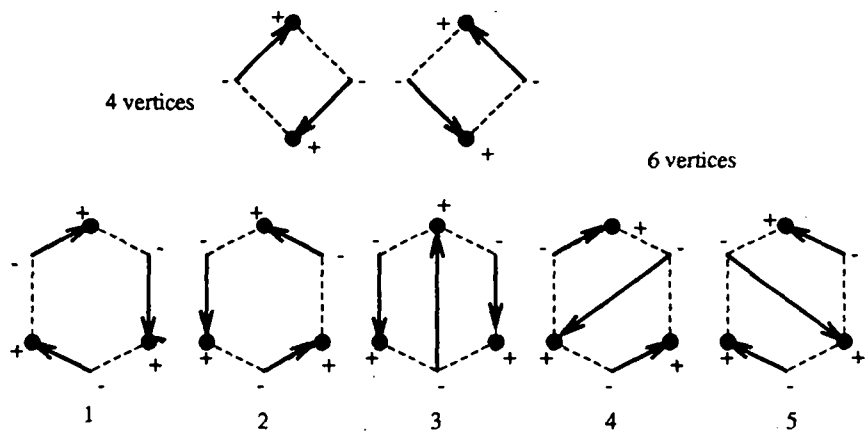


Figure 8: Links between the vertices of the 3D polygon

a list of oriented segments. The oriented segments are based on the points $\{(Q_i, L_i)|i \in [0..n]\}$ produced by the application of *CYCLESTOPOINTS*. It can be implemented in a various number of ways, so far that each point Q_i is used once, and only once, in the constitution of the oriented segments, that each segment starts with a point Q_i whose label L_i is $-$, and ends at a point whose label is $+$. The last condition is about the planar graph which constitute the 3D polygon and the new segments. It must be such than if the 3D polygon is drawn on a plane as a convex polygon, no couple of segments produced by *CYCLETOSEGS* cross each other. Figure 8 illustrates this last property for the cases with 4 and 6 vertices, which ensures that the 3D reconstructed curve doest not intersect itself.

Let see one way to achieve this : $\{(Q_i, L_i)|i \in [0..n]\}$ is not only an ordered set of points, it can be considered as an oriented cycle of points, that is, the last element can be removed and put in the first place, therefore, we can ensure that the first point has a label $-$. Then we take those segments 2 by 2 in the order of the list, and we constitute oriented segments from $-$ to $+$. We can see that we could have also chosen the other way round (case 1 and 2 of figure 8). This has no influence on the completeness of the reconstructed curves. We will discuss much of these possibilities later on.

As we can see, *CYCLETOSEGS* is a generalization of the procedure *BUILDSEGMENTS* to the case of polygons with an arbitrary number of

vertices. We can build now a procedure *CELLTOSEGS*, which generates a given number of oriented segments (at most 12) for each cell of the 3D grid.

theorem 7 *The application of* CELLTOSEGS *to the entire set of the cells of the 3D grid produce a set of oriented segments which constitute oriented, not self intersecting, and complete (i.e. closed) 3D curves, except at the boundary of the grid.*

Proof: Once again, the proof looks like the one of theorem 3. Let Q be the end point of one oriented segment produced by *CELLTOSEGS*, which is not at the boundary of the 3D grid. Q is strictly included in the face of two adjacent cells. Q is also the interpolated position along a segment $]P1, P2[$ of two 3D polygons representing the iso-surface of f in the 2 adjacent cells, the orientation of this segment being opposite for the two 3D polygons. Therefore, Q has a label + in one of the cell, and a label − in the other one. There is therefore exactly one incoming oriented segment in the cell +, and one outgoing oriented segment in the cell −, which proves that the reconstructed curves are oriented and complete (except at the boundary of the image). The "not self intersecting" properties is directly due to the last condition required for *CYCLETOSEGS*.

Furthermore, by construction, the orientation of the reconstructed bi-iso-lines is compatible with the *bi-iso-line* orientation convention. Once again, what is remarkable is that *local* computations can ensure *global* topological properties.

4.3 Variants

A first class of variants is about the computation of the interpolated points Q_i, a second class is variations about the implementation of the *CELLTOSEGS* procedure.

Notice that taking (g, f) doesn't lead to the same extracted 3D curves, with oposite directions, than when taking (f, g). This is due not only to the arbitrary choice of the *bi-iso-line* orientation convention, but also to the non-commutativity of the linear interpolation first along the cell side, then along the 3D polygon side of g. If this symetry property is required, a simple way to achieve this is to replace the computation of the interpolated possitions of the points (Q_i, L_i) with the following computation :

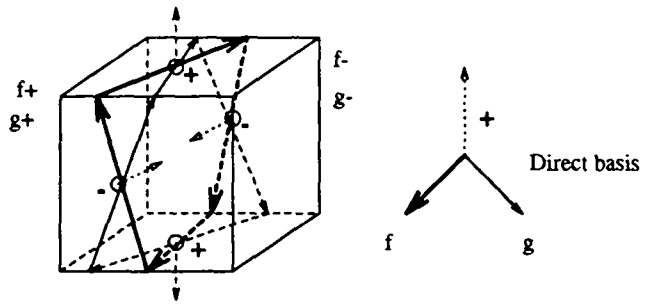


Figure 9: Intersection of two cycles around a voxel

To each face of the cell, we apply *BUILDSEGMENTS* with the values of f, which gives 0, 1 or 2 oriented segments, we apply also *BUILDSEGMENTS* to the values of g, which gives also 0, 1 or 2 oriented segments, and then to consider every couples of segments $(\vec{s}_1, \vec{s}_2)$, one from f and the other from g. Great care has to be taken for degenerated cases, and solved accordingly to the *bi-iso-line* orientation convention (see figure 6). When the case is not degenereted, we compute the intersection point Q of those two segments (if any) and we give it a label L which is the sign of the basis $(\vec{s}_1, \vec{s}_2, \vec{n})$ (i.e. sign of $(\vec{s}_1 \wedge \vec{s}_2) \cdot \vec{n}$), where $\vec{n}$ is the normal of the cell face, pointing toward the outside (see figure 9, and also figure 6). This convention is nothing else than another expression of our *bi-iso-line* orientation convention. This times the roles of f and g are really anti-symetric.

Another remark is that the points Q_i are not the intersections of the iso-contours defined by the tri-linear interpolation with the faces of the 3D cells. If this property is requiered, the solution consists in replacing the segments intersection solution previously described with the intersection of the hyperboles corresponding to the bi-linear interpolation of f and g on that face (see [16]).

4.4 A word about the procedure *CYCLETOSEGS*

A voxel where two iso-patches intersect in more than two points is a singularity for the topology of the reconstructed bi-iso-lines. This is reflected by the choice that we have for the implementation of the function *CYCLETOSEGS*.

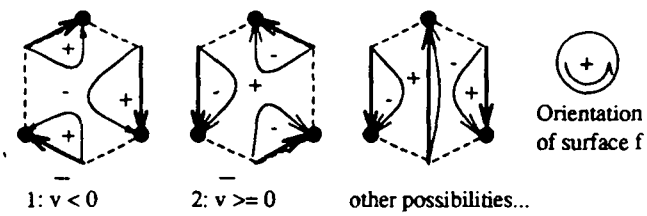


Figure 10: Bi-iso-lines evolution with a small perturbation of almost parallel iso-surfaces

For those points, we could say that we have not enough information to determine the topology of the intersection lines (see figure 10). We can decide to choose arbitrarily between the solutions (the choice is local, it doesn't depend on neighboring voxels and doesn't interfere with the continuity property of the solution). We propose the use of the following convention to decide between the two implementations that we proposed for *CYCLETOSEGS*. Remember that the set of Q_i constitutes also a cycle of points on the iso-surface of f, which can be oriented one way or the other.

convention 4 (The *average value* **orientation convention)** *The cycle* $\{Q_i\}$ *is oriented such that the label of the inside region of this cycle, on the oriented iso-surface of* f *and given by the application of the* left handed *convention is the same than the label given to* $\bar{v}$ *by the application of the* object/background *convention.* $\bar{v}$ *being the average value of the values associated to the points* $\{Q_i\}$.

Cases 1 and 2 of figure 10 take this convention into account. We think that this convention will be useful when we will be to consider the 4D or ND cases, because then the computation of the oriented segments would need to be deterministic for the whole image, in the same way than *BUILDSEGMENTS* must be deterministic when going from 2D to 3D. In fact, we used then this very same convention (applied to $\hat{v}$, instead of $\bar{v}$).

5 Applications

The Marching Lines algorithm can be applied with any couple of images or implicit equations. It can be also implemented in the randomized way : some seeds are choosen at random in the 3D grid, and if ever one bi-iso-line come through this cell, then we follow the line "marching" from one voxel to the other one. We have implemented both the randomized and the extensive computation of bi-iso-lines. We have shown that with very few seeds, the lines are extracted roughly in the order corresponding to decreasing length, which can be used to tremendously reduce the number of computation, for example in the case of our registration algorithm based on crest lines (see [17]).

In a previous paper (see [16]), we have shown how to compute the differential characteristics of the iso-surfaces, directly from the values of a 3D image, using linear filtering to compute the image derivatives. Those differential values can be the gaussian curvature K and the average curvature S of the iso-surface, the principal curvatures k_1 and k_2 and the associated principal directions $\vec{t_1}$ and $\vec{t_2}$, and also a criterion e, called the "extremality" (introduced in [11]) whose zero-crossings characterize the points belonging to the crest lines. The crest lines are 3D lines of the surface where the curvature is locally maximal, more precisely, where the maximal curvature is extremal in the corresponding principal direction.

We have shown that those functions can be computed for almost any point of the 3D image where the gradient is not 0. Each of these 3D functions can be used to define the second implicit surface $g(x, y, z) = J$, whose intersection with an iso-surface f of the 3D image defines the characteristic lines. The Marching Lines Algorithm is the canonic tool to extract automatically those 3D curves. For example, $K = 0$ defines lines which are refered to as the parabolic lines in the field of the Differential Geometry. In the same way, we can extract any "iso-gaussian" curve, corresponding to a constant value of the gaussian curvature K (see figure 11). The best application that we found is the extraction of crest lines, and their use to the automatic registration of 3D images (see [17], [3]). Unfortunately, e is not defined everywhere, for example for umbilic points, and in that case, the reconstructed lines are not closed (see [16]). $e = 0$ defines the crest lines, corresponding to the successive loci of points of the surface where the curvature is localy maximal. Figures 12 and 13 show respectively the maximum and minimum curvature

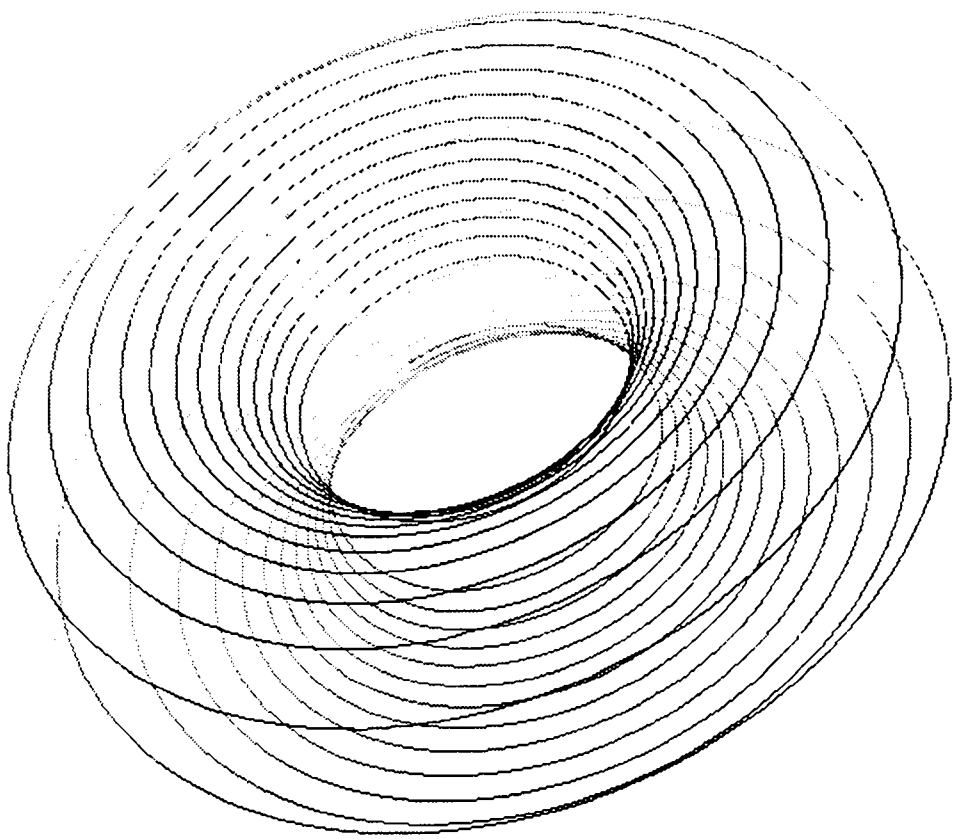


Figure 11: Iso-gaussian curves for the synthetic example of the 3D image of a torus, and for regularly spaced values of the gaussien curvature. The algorithm ensures that the lines are closed.

lines, extracted with the Marching Lines from a 3D synthetic image. Figure 14 are the crest lines extracted from two different 3D scans of the same patient, taken at two different times, and figure 15 is the superimposed lines after automatic registrations. We can see that those lines are very stable and reliable landmarks. Furthermore, those crest lines, which are defined as differential geometry features, appear to be anatomical features in the case of medical images. Figure 16 shows the crest lines extracted from a 3D Magnetic Resonance Image, which correspond to the crests of the circumvolution of the brain, called "Gyri" by the physicians. We espect to use those lines to automatically label 3D images, with the help of an atlas of crest lines.

6 Conclusion

To conclude, we have described and proved a very powerful and general-purpose tool for the automatic extraction of 3D curves from 3D images, called the Marching Lines algorithm. It is powerful because it ensures good topological properties of the reconstructed lines, and general-purpose because it only requires as input two images, or two continuous functions and two iso-values thresholds. Along with the conputation of the differential characteristics of iso-surfaces, that we have presented in a previous paper, a large variety of feature lines can be computed, such as parabolic lines, iso-gaussian curves, and also crest lines. This last application is very promizing for the automatic registration of 3D image, and perhaps also for the automatic labeling of anatomical features. We hope that those new tools will be used in many other ways, to compute 1D characteristic features out of 3D images, and also as primitive for higher level image processing algorithms, such as matching or pattern recognition, based on lines.

Acknowledgment

We want to thank Olivier Monga, Serge Benayoun and Nicholas Ayache for stimulating discussions about crest lines, and also to Janes Wilhelms and Allen Van Gelder for discussions about iso-surfaces. We want to thank André Gueziec for his 3D line registration software. We want also to thank Digital Equipment Corp. who provide us with fast computers, GE-CGR of Buc, France, who provided the two scanner images of the head phantom. Bob

Hummel deserves special thanks for his careful review of a previous version of this paper.

References

[1] M.J. Düurst. Additional references to marching cubes (letter). *Computer Graphics: a Quaterly Report of SIGGRAPH-ACM*, 22:72–73, 1988.

[2] R. S. Gallagher and J.C. Nagtegaal. An efficient 3-d visualization technique for finite element models and other coarse volumes. In *Computer Graphics*, volume 23(3), pages 185–194, 1989.

[3] A. Guéziec and N. Ayache. Smoothing and matching of 3D-space curves. In *Proceedings of the Second European Conference on Computer Vision 1992*, Santa Margherita Ligure, Italy, May 1992.

[4] G.T. Herman. On topology as applied to image analysis. *Computer Vision, Graphics, and Image Processing*, 52:409–415, 1990.

[5] G.T. Herman and D. Webster. A topological proof of a surface tracking algorithm. *Computer Vision, Graphics, and Image Processing*, 23:162–177, 1983.

[6] Alan D. Kalvin. A survey of algorithms for constructing surfaces from 3d volume data. Technical Report RC 17600, IBM Research Division, January 1992.

[7] Jan J. Koenderink. *Solid shape.* The MIT Press, 1990.

[8] T.Y. Kong and A. Rosenfeld. Digital topology: introduction and survey. *Computer Vision, Graphics, and Image Processing*, 48:357–393, 1989.

[9] Michael Laszlo. Fast generation and display of iso-surface wireframes. *CGVIP: Graph. Models and Im. Proc.*, 54(6):473–483, November 1992.

[10] Wiliam E. Lorensen and Harvey E. Cline. Marching cubes: A high resolution 3d surface reconstruction algorithm. *Computer Graphics*, 21(4), July 1987.

[11] Olivier Monga, Serge Benayoun, and Olivier D. Faugeras. Using partial derivatives of 3d images to extract typical surface features. In *Proceedings CVPR '92, Urbana Champaign, Illinois.* IEEE, July 1992. also an INRIA Research Report (1599).

[12] B. A. Payne and A.W. Toga. Surface mapping brain function on 3d models. *IEEE Computer Graphics and Applications*, 10(2):41–53, 1990.

[13] Azriel Rosenfeld, Yung T. Kong, and Angela Y. Wu. Digital surfaces. *CGVIP: Graph. Models and Image Processing*, 53(4):305–312, July 1991.

[14] J-P Thirion. The gradient and laplacien filtered back projection operators. Research report (submitted to IEEE TMI) in press, INRIA, October 1992.

[15] J-P. Thirion, N. Ayache, O. Monga, and Gourdon A. Dispositif de traitement d'informations d'images tri-dimensionnelles avec extraction de lignes remarquables. Brevet Français, numero 92 03900, Mars 1992. Pattent pending.

[16] J-P. Thirion and A. Gourdon. The 3d marching lines algorithm and its application to crest lines extraction. *rapport de recherche INRIA*, (1672), May 1992.

[17] J-P Thirion, O. Monga, Benayoun S., Gueziec A., and Ayache N. Automatic registration of 3d images using surface curvature. In *IEEE Int. Symp. on Optical Applied Science and Engineering*, San-Diego, July 1992.

[18] Ake Wallin. Constructing isosurfaces from ct data. *IEEE Computer Graphics and Applications*, pages 28–33, November 1991.

[19] Jane Wilhelms and Allen Van Gelder. Topological considerations in isosurface generation. Technical Report UCSC-CRL-90-14, CIS Board, University of California, Santa Cruz, 1990. Extended abstract in ACM Computer Graphics 24(5) 79–86.

[20] G. Wyvill, C. McPheeters, and C. Wyvill. Data structures for soft objects. *Visual Computer*, 2:227–234, 1986.

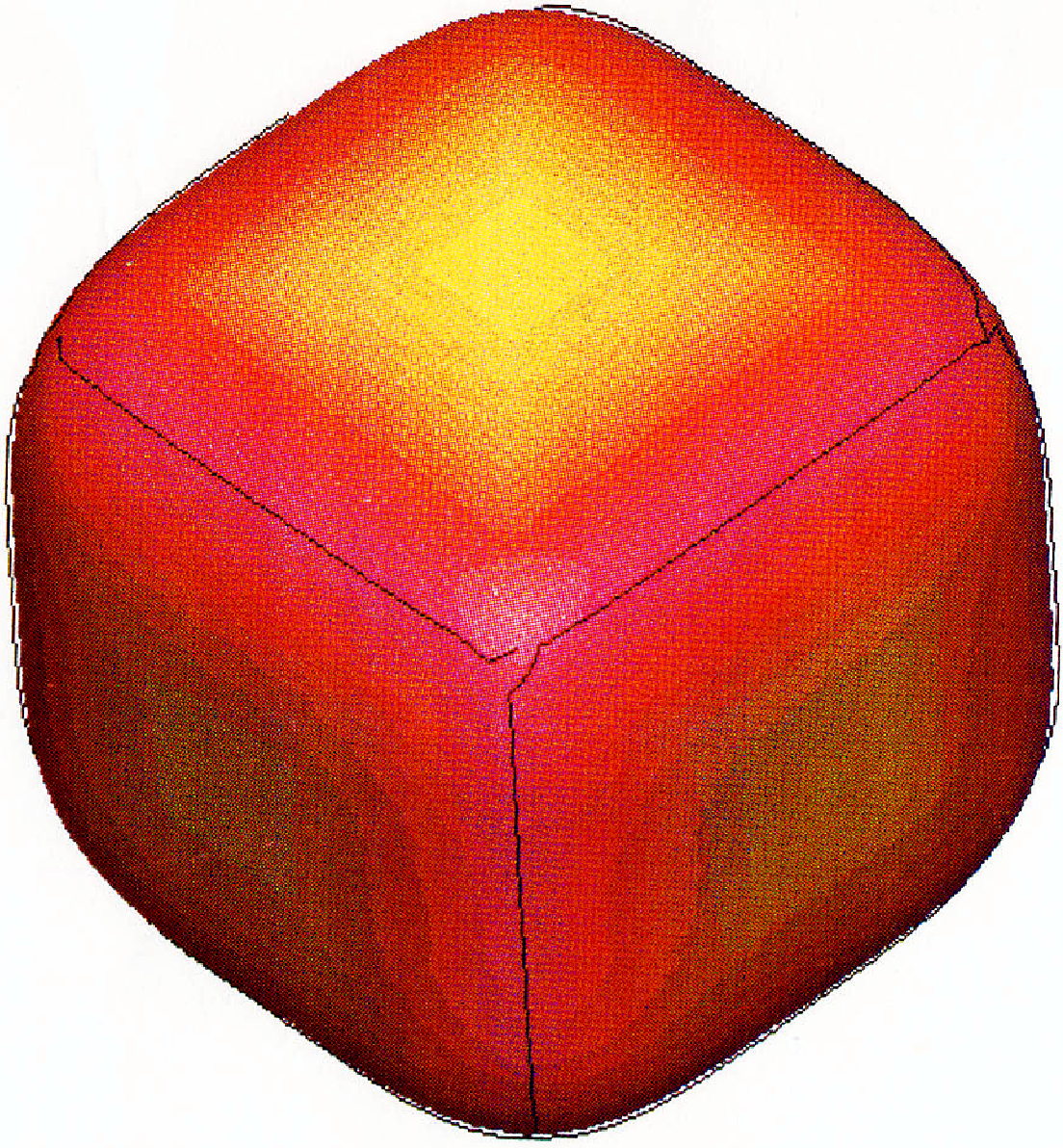


Figure 12: Iso-surface of the 3D image of a cube function, colored with the maximal curvature values, and crest lines, extracted with the Marching Lines algorithm. Lines are not closed because of the umbilic points.

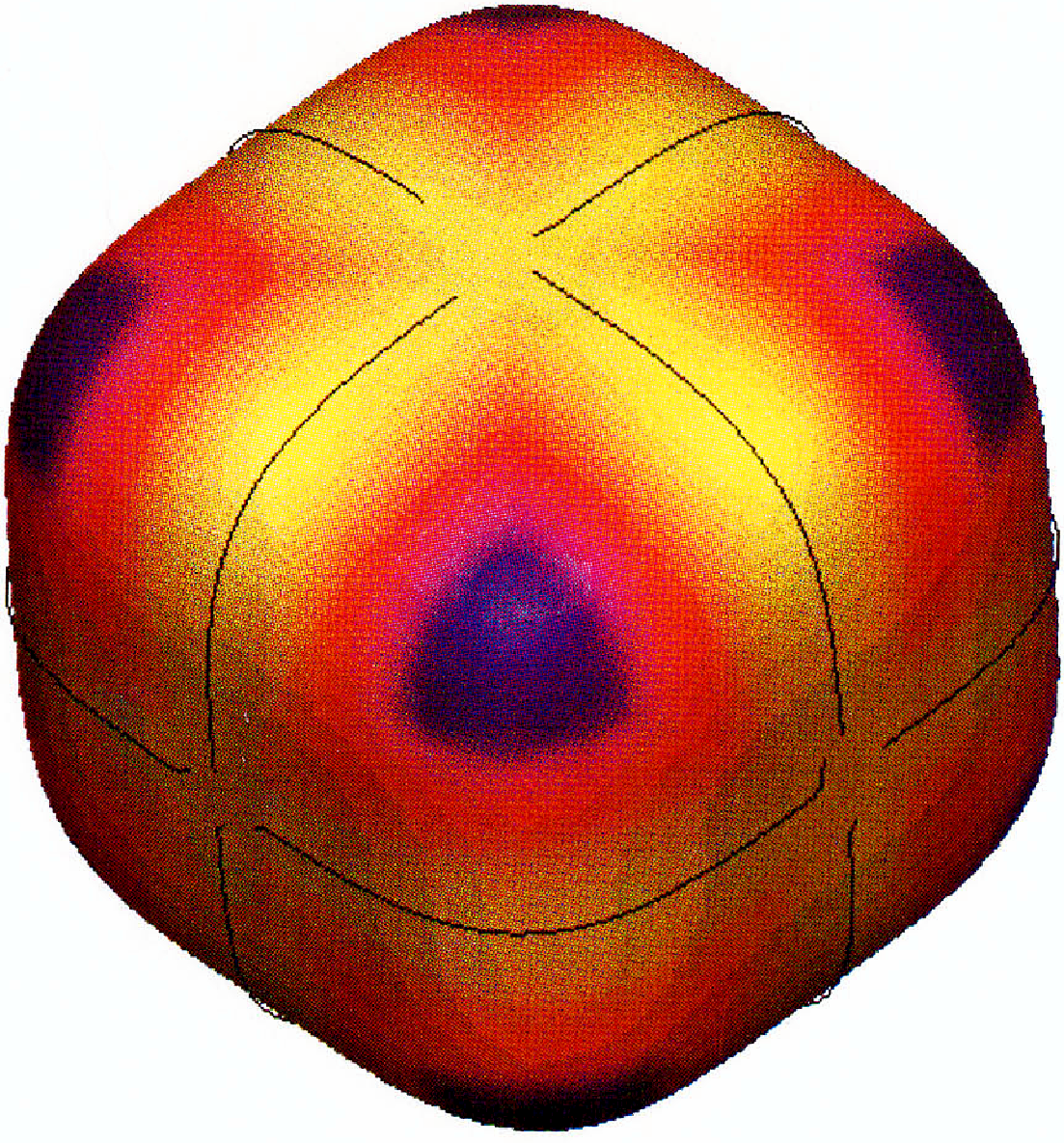


Figure 13: Iso-surface of the 3D image of a cube function, colored with the lesser of the two principal curvatures, and lines of minimal curvature, extracted with the Marching Lines.

The 3 following figures are the same as in the previous paper "Computing the Differential Characteristics of Iso-Intensity Surfaces", figures 10, 11, 12.

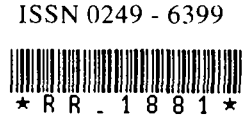
Figure 14 : Crest lines for two 3D XRay scanner images, with resolution 160x200x137. The width of the lines is sub-voxel (about 1/5th)

Figure 15 : 3D automatic registration of the two crest line sets (red and blue), which are sometimes exactly superimposed. This proves the high quality and stability of the extracted features.

Figure 16 : Crest lines corresponding to the circumvolutions of the brain, extracted automatically from a 3D Magnetic Resonance Image. Most of those lines have anatomical names.

Unité de Recherche INRIA Sophia Antipolis
2004, route des Lucioles - B.P. 93 - 06902 SOPHIA ANTIPOLIS Cedex (France)

Unité de Recherche INRIA Lorraine Technopôle de Nancy-Brabois - Campus Scientifique
615, rue du Jardin Botanique - B.P. 101 - 54602 VILLERS LES NANCY Cedex (France)
Unité de Recherche INRIA Rennes IRISA, Campus Universitaire de Beaulieu 35042 RENNES Cedex (France)
Unité de Recherche INRIA Rhône-Alpes 46, avenue Félix Viallet - 38031 GRENOBLE Cedex (France)
Unité de Recherche INRIA Rocquencourt Domaine de Voluceau - Rocquencourt - B.P. 105 - 78153 LE CHESNAY Cedex (France)

EDITEUR
INRIA - Domaine de Voluceau - Rocquencourt - B.P. 105 - 78153 LE CHESNAY Cedex (France)

ISSN 0249 - 6399