The 3D marching lines algorithm and its application to crest lines extraction

Jean-Philippe Thirion, Alexis Gourdon

UNITÉ DE RECHERCHE
INRIA-ROCQUENCOURT

Institut National
de Recherche
en Informatique
et en Automatique

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

Rapports de Recherche

N°1672

Programme 4

Robotique, Image et Vision

THE 3D MARCHING LINES ALGORITHM AND ITS APPLICATION TO CREST LINES EXTRACTION

Jean-Philippe THIRION
Alexis GOURDON

Mai 1992

The 3D Marching Lines Algorithm and its Application to Crest Lines Extraction

Jean-Philippe THIRION, Alexis GOURDON
INRIA, Epidaure *

April, 1992

Abstract

This paper presents a powerful and general purpose tool designed to extract characteristic lines from 3D images [1]. The algorithm, called *Marching Lines*, is inspired from the Marching Cubes algorithm, which is used to extract iso-value surfaces from 3D images. The Marching Lines algorithm extracts with sub-pixel accuracy the 3D lines corresponding to the intersection of two iso-surfaces from two different 3D Images. We show how to implement this algorithm to ensure that the reconstructed 3D lines have good topological properties, mainly that they are continuous and closed. We also present a new method to compute the differential characteristics of iso-surfaces, and show an application to the extraction of crest lines in 3D Images. We explain that a crest line can be locally defined as the intersection of two surfaces, one corresponding to an iso-value in the image, and the other one to a *crest surface* which we define in this paper, and whose implicit equation can be directly computed from the voxel values of the 3D image. Finally, experimental results for 3D images of the skull are presented, where crest lines are extracted, and used to compute automatically the geometric transform between two 3D scanner images of the same subject, taken in two different positions.

Key words: 3D Image, Segmentation, Crest Line, Marching Cubes, Topology, Differential Geometry.

*http://zenon.inria.fr:8003/Equipes/EPIDAURE-eng.html

[1] This process is part of a patent taken out by the INRIA

L'algorithme du "Marching Lines" et son application à l'extraction des lignes de crête.

Jean-Philippe THIRION, Alexis GOURDON
INRIA, Epidaure

Résumé :
Cet article présente un outil nouveau et puissant, conçu pour extraire des lignes caractéristiques à partir d'images tri-dimmensionnelles [2] . Cet algorithme, appelé *Marching Lines*, est inspiré de l'algorithme du *Marching Cubes* qui sert pour sa part à extraire des surfaces d'iso-valeurs d'images 3D. Le *Marching Lines* permet d'extraire, avec une précision supérieure au pixel, les lignes 3D correspondant à l'intersection de deux iso-surfaces, obtenues à partir de deux images 3D différentes. Nous montrons comment implémenter l'algorithme de manière à garantir que les lignes 3D reconstruites ont de bonnes propriétés topologiques; principalement que ces dernières sont continues et fermées. Nous présentons également dans le même article une façon nouvelle de calculer les caractéristiques différentielles des iso-surfaces, directement à partir de l'image 3D, et nous en montrons l'application à l'extraction des lignes de crête. Nous expliquons qu'une ligne de crête peut être définie localement comme l'intersection de deux surfaces, dont l'une correspond à une iso-surface de l'image et l'autre à une *surface de crête* que nous définissons dans cet article, et dont l'équation implicite peut être directement calculée à partir des valeurs des voxels de l'image 3D. Enfin, nous présentons quelques résultats expérimentaux pour deux images scanners 3D d'un crâne, prises dans deux positions différentes, et où les lignes de crête sont extraites et utilisé es de manière automatique pour retrouver la transformation géométrique existant entre les deux images 3D.

Mots clef: Image 3D, Segmentation, Ligne de crête, Marching Cubes, Topologie, Geometrie differentielle.

[2] Ce procédé fait partie d'une demande de brevet déposée par l'INRIA

1 Introduction

Image processing 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.

In the first sections, we present a general purpose tool called Marching Lines, which can be used to extract lines defined by the intersections of two iso-surfaces. Our aim is to give some guarantees about the topological properties of the reconstructed curves. In the second part, we present a new method to compute the differential characteristics of the iso-surfaces from the derivatives of the 3D image, such as principal curvatures and principal directions. Then we explain how to use this method, along with the marching line algorithm, to extract crest lines. Finally, we present experimental results of crest line extraction with the 3D scanner image of a head.

2 Improving the Marching Cubes algorithm

Generally, a 3D image comes from some measurement performed on a given object. It can be for example a Magnetic Resonance Image (MRI), a 3D Xray scanner image, or a Geological Image. In this paper, we are not concerned with the extent to which a digital image reflects the shape of the objects that are scanned. Approaches to this problem can be found in [1], [2], and [12] for 3D images. In the present paper, we are only interested in iso-surfaces, defined as the boundaries between regions whose intensity is higher or equal to a given threshold, and regions whose intensity is lower. A survey of iso-surface techniques can be found in [14] or in [8]. Fortunately, iso-surfaces typically correspond to object boundaries in medical image applications, because the voxel values reflect directly the density of matter at that point inside the body, and anatomical objects such as bones have homogeneous density. Furthermore, medical images are the major source of 3D

images. For any infinite continuous 3D image, iso-boundaries define continuous, non-intersecting surfaces, without hole. A Klein bottle, for example, cannot be defined as an iso-surface, but a torus can be. Real images are finite, therefore an iso-surface is still without holes, except when it intersects the boundary of the image (which defines a closed, non-self-intersecting contour). Real images are also sampled, generally on a regular 3D grid. In this paper, we will assume that a voxel is a cube of the 3D space, and that the values of the 3D image correspond to the values of the intensity function at each vertex of the cube. To avoid ambiguities, iso-surfaces can be defined by the continuous image obtained from the convolution of the digital image with an interpolation function. Iso-surfaces are therefore entirely defined with the digital image, the interpolation function and the threshold of iso-value. If we set a fixed value for each voxel of the image (interpolation of order zero), say for example the mean of the eight vertices, then the iso-surfaces can be considered from the view point of discrete topology [7], [9], [6]. The Marching Cubes algorithm corresponds, to some extent, to the use of the linear interpolation.

To summarize the Marching Cubes algorithm, as defined by Lorensen and Cline in [10], the eight vertices of the voxel can be labeled with '$-$' or '$+$', according to the value of the vertex with respect to the selected iso-value. This defines a 8-bit code (that is, 256 cases) for each voxel. A 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 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 are without holes (first noticed in [4]). For example, with the notation used in [10], the cube of Case 3, with '$-$' and '$+$' exchanged, put on top of the cube of Case 10, leads to holes in the reconstructed surface. In the following, we propose another solution which ensures that the surfaces have no holes (except for image boundaries). A former solution to that problem has been proposed in [13]. It is based on a partition of the voxel into five tetrahedra, but this solution requires more triangles to describe the reconstructed surfaces, and is not isotropic. Our solution is closer to the tri-linear interpolation definition of the iso-surfaces.

2.1 The 2D case

In order to decompose the problem, we will first study the 2D case. Labeling each vertex with respect to the iso-value leads to 16 cases, which can be reduced to seven with symmetry considerations (Figure 1, see also [15]). The iso-lines defined with the linear interpolation correspond to hyperbolas passing through each pixel. The two branches of each hyperbolas are separated with asymptotic lines which are *isothetic* (parallel to the major axes). We approximate these branches of hyperbolas with segments, intersecting the hyperbolas at the edges of the pixel. Our algorithm corresponds to the linear interpolation only "to some extent" because of this approximation. This segmentation is entirely defined, except when the sign of the vertices is alternatively '+' and '−'.

One way to choose between Case 6 and Case 7 of Figure 1 is to compute the average value $\overline{v}$ of the four vertices and compare it with the iso-value threshold I. If $\overline{v} \geq I$ we choose Case 7 else if $\overline{v} < I$ we choose Case 6. In all case, the reconstructed segments are oriented with respect to the position of the '−' and '+' vertices, such that, according to the orientation, the region labeled '+' lies on the left side of the curve (see "Convention" in Figure 1). 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. The exact solution (with respect to the topology of the bi-linear interpolation) is to compute the value of the bi-linear interpolation at the crossing point of the two asymptotes of the hyperbola, which corresponds, with the notation of Figure 2, to test the value $\hat{v} = (s_1 s_4 - s_2 s_3)/(s_1 + s_4 - s_2 - s_3)$ with the iso-value I.

The proof that the reconstructed lines are continuous and closed (except for image boundaries) is obvious. If the degenerated case expressed in figure 2 does not occur, the reconstructed lines define objects which have the same connectivity as the iso-boundaries defined with the linear interpolation.

The case $\overline{v} = I$ has to be considered with special care (see Fig.2). 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). It corresponds to the case when there is only a singular point of contact between two parts of an object. This corresponds to connectivity in classical topology, but non-connectivity if we consider that two objects are connected only if a strip of positive width can be found between the two parts. Roughly, a closed set is

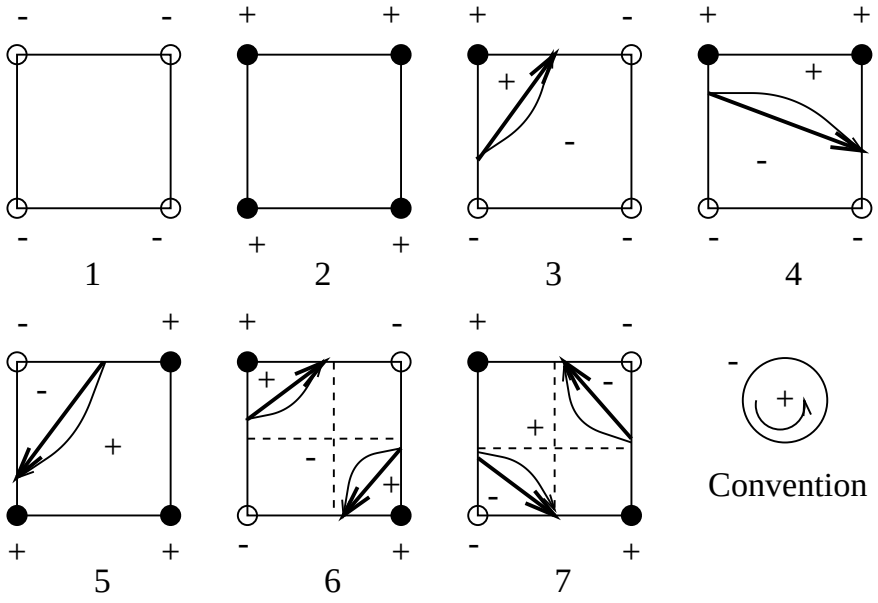

Figure 1: Iso-boundaries for the 2D case

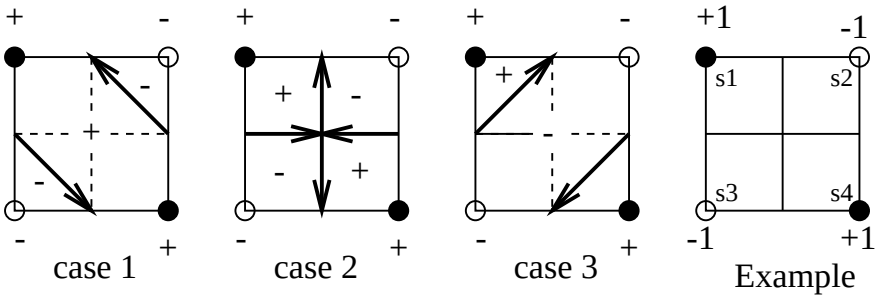

Figure 2: Iso-boundaries: singular situation

strip-connected if there is a path in the set joining any pair of points of the interior of the set (arc connectivity), which lies in the inside of the set. That is, strip connectivity implies that there is a neighborhood around each point of the path, contained entirely within the set, which defines a strip. Case 1 in Figure 2 leads to the strip connectivity of the '+' region, Case 2 keeps the singularity, and Case 3 leads to the strip connectivity of the '−' region. In order to obtain the good topological properties of the iso-surfaces, we must choose, once and for all, for the entire image, between Case 1 or Case 3 in the singular situations. This correspond to the choice of one of the two following definitions of an iso-surface: "an iso-surface is the surface which separates regions with $i \geq I$ from regions with $i < I$" or "which separates regions with $i > I$ from regions with $i \leq I$". Therefore, an arbitrary choice has to be done for continuous topology in the same way as for discrete topology, when 4 or 8 connectivity is chosen.

2.2 The 3D case

In 3D, the linear interpolation defines cubic surfaces whose intersection with isothetic planes are hyperboles. This cubic surface is composed of several disconnected patches. We call each of these connected patches an *iso-patch*. Our method has good topological properties because it reflects the properties of the underlying iso-patches.

Let's see how those iso-patches can be approximated (Figure 3). We orient each side of the voxel toward the exterior of the cube, then we consider each side in the same way as for the 2D case, which leads to a segmentation of each face. If we have no singular cases, such as described in Figure 2, it is very easy to show that those segments define closed oriented loops around the cube (we are ensured that the end point of a segment is the starting point of another segment). Those loops correspond to 3D faces which are not always planar polygons (Figure 3). We will refer to these faces, which are bordered by 3D polygonal curves, as 3D polygons. Examples 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 triangulations. Finally, we can see in example 3 that the case with 12 vertices can occur.

We can associate a piece of surface to each polygon, such as a triangulated

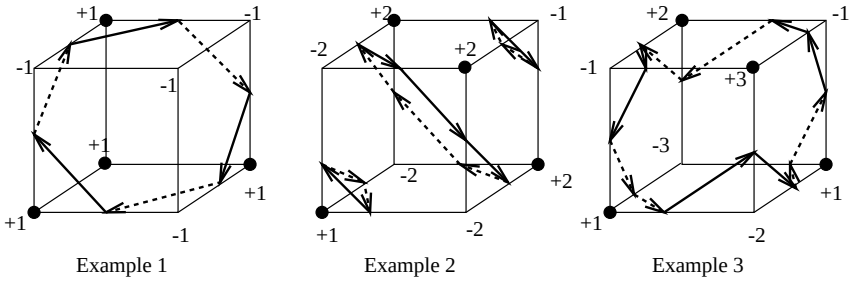

Figure 3: Examples of cycles of iso-surfaces around a voxel

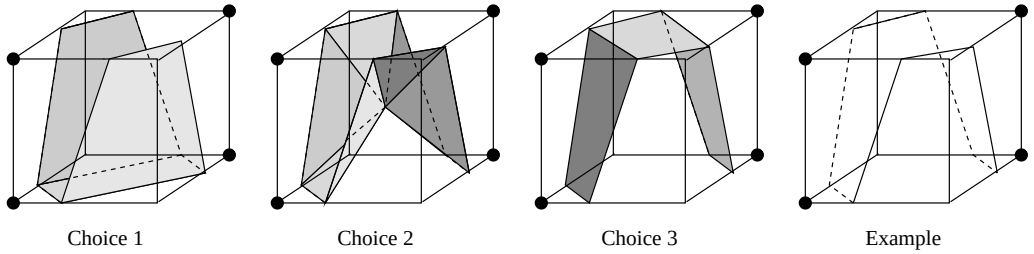

Figure 4: Different choices for the triangulation

surface whose edges are the edges of the polygon, or the minimal surface having the polygon as boundary. When the number of vertices is greater than 3 the surface can be triangulated in many different ways, as can be seen in Figure 4. We suggest triangulating the polygon with respect to its barycenter (Choice 2).

We are ensured that the reconstructed surfaces exactly interpolate the iso-patches at each vertex of the 3D polygon (which corresponds to its intersection with the edges of the voxel). The reconstructed surface has all the requisite properties: continuity, absense of holes, and the same connectivity as the iso-surfaces defined with the tri-linear interpolation (if $\hat{v}$ is used). The fact that the reconstructed surface has no holes doesn't mean that the volume defined with this surface has no holes; in fact the volume can be a torus or a volume of any *"genus"* (but of course not a Klein bottle).

To conclude, our algorithm allow us to compute a triangulated surface

which has the same good global topological properties as the underlying iso-surfaces, even if the computation is performed only locally and independently for each voxel. We conjecture also that the same algorithm can be extended to Euclidean spaces of dimension $n > 3$, using for each hyperside of a cell the segmentation defined for dimension $n - 1$.

3 The Marching Lines algorithm

Now that we are ensured that the reconstructed iso-surfaces have good topological properties, let consider how to compute the intersection of two iso-surfaces. We will call the corresponding lines *bi-iso-lines*, ("bi" in order not to mistake them with iso-boundaries in 2D, and refering to the pair of surfaces). The bi-iso-lines are entirely defined by two 3D digital images, two corresponding smoothing functions, and two iso-values. 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. Another solution, very simple and convenient, is to use the decomposition into tetrahedra of the voxels as proposed by Payne and Toga in [13]: we have at most one iso-surface within each tetrahedron for each image. The intersection of the two iso-surfaces is defined without ambiguity and the reconstructed bi-iso-lines have once again good topological properties: they are continuous, closed curves. We will develop in this section another solution, closer to the definition of the iso-surfaces based on the tri-linear interpolation of the voxels values. All results concerning the orientation of the bi-iso-lines as found by our algorithm as presented in this section would also be true for an algorithm based on the tetrahedra decomposition method.

3.1 Orientation considerations

Consider the side of a voxel, oriented toward the outside of the cube. We can compute the segmentation of this side in the same way as defined in the previous section, this time for two different 3D images (say Image 1 and Image 2). The segments of Image 1 can intersect the segments of Image 2 for at most four points per side, except when the two segments are superimposed. In fact, because the branches of the hyperboles are separated with isothetic asymptotes, there are at most two intersections per side. A

convenient solution to avoid the degenerated cases is to slightly perturbate the data. To avoid the superimposition of two segment, or the fact that two segments start from the same point, we systematically displace the vertex of a cycle of Image 2 by a small distance along the corresponding edge of the voxel, when it exactly corresponds to an vertex of a cycle of Image 1. Remember that, in order to guarantee the continuity, the computation of the vertices must be independent of the voxel for which they are computed. For example, we choose to perform this displacement along the edge of the voxel always in the direction of increasing coordinates. In the following, we will suppose that these degenerate cases have been solved.

Let's consider one of these intersection points. Recall that the segments on the side of the voxel are oriented. If $\vec{v_1}$ represents the oriented segment of Image 1 and $\vec{v_2}$ the segment of Image 2, the intersection point can be labeled '$-$' or '$+$' according to the direction of the cross-product $\vec{v_1} \wedge \vec{v_2}$ with respect to the normal of the side of the cube (pointing toward the exterior, see Figure 5). With this convention, if there are two distinct intersection points on the side, then their labels are opposite. Let's now take two oriented 3D polygons P_1 and P_2 as defined in the previous section, corresponding to two iso-patches of Image 1 and Image 2. The number of intersection points between P_1 and P_2 is necessarily even and the labels associated to those points are alternatively '$+$' and '$-$'. Therefore, the intersections of P_1 and P_2 define another 3D polygon P, with an even number of vertices, labeled alternatively '$+$' and '$-$', as can be seen in the example of Figure 5.

In order to approximate the bi-iso-lines, we draw segments between pairs of points belonging to the same 3D polygon P, which ensures that those points belong to the same two iso-patches of the two different images. The segments are drawn from the points labeled '$-$' to the points labeled '$+$' (figure 6). The most common case is when P has only two vertices, which defines a single segment that we orient from '$-$' to '$+$'. If P has four, six or more vertices, then several solutions can be chosen. The fact that two bi-iso-lines cannot cross one another in either of the iso-surfaces tells us that linking the vertices of P must be done such that if the vertices of P are numbered and projected on a plane to form a convex polygon, then there is no crossing between two segments. This leads to two solutions for four vertices and five solutions for six vertices (see Figure 7).

If fact, the number of choices increase in the case when the two iso-surfaces are almost tangent. The intersection is hard to define then, and a

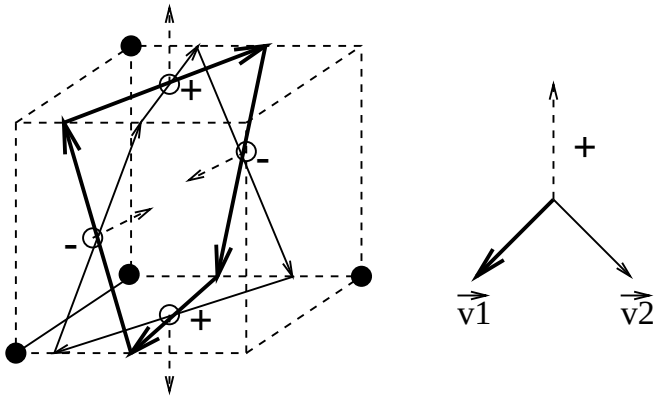

Figure 5: Intersection of two cycles around a voxel

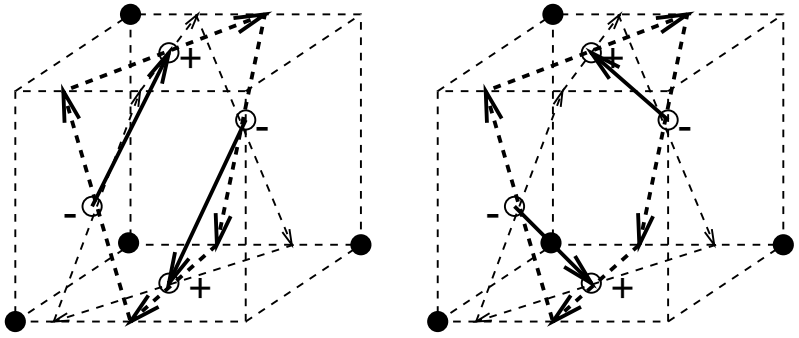

Figure 6: Choice between the vertices, 3D view

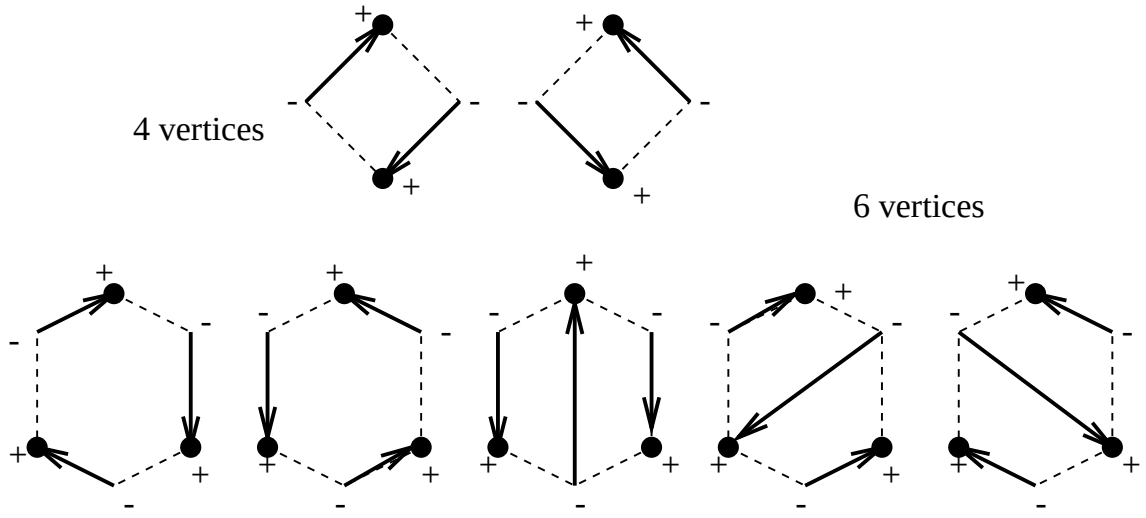

Figure 7: Links between the vertices of the 3D polygon

small perturbation of the data can lead to a swift change in the topology of the intersection lines, as can be seen in Figure 8.

A voxel where two cycles intersect in more than two points is a singularity for the topology of the reconstructed bi-iso-lines. For those points, we could say that we have not enough information to determine the topology of the intersection lines. We can decide to choose arbitrarily between the solutions (the choice is local, it doesn't depend on neighboring voxels and doesn't

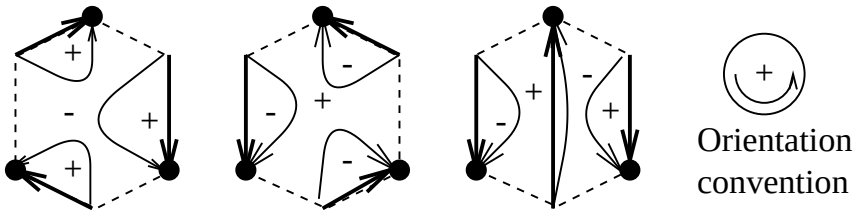

Figure 8: Bi-iso-lines evolution with a small perturbation of almost parallel iso-surfaces

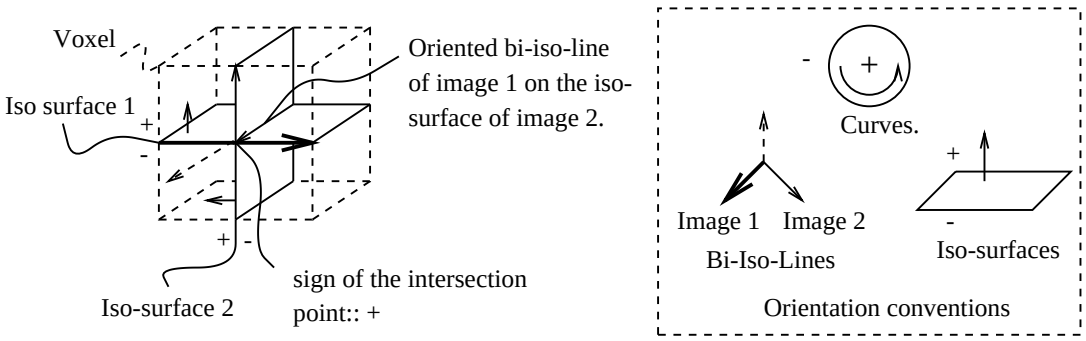

Figure 9: Orientation conventions for the bi-iso-lines

interfere with the continuity property of the solution). Heuristics can also be used, based on the average value of the intensities at the vertices of P. Each of the arbitrary solutions will lead to reconstructed bi-iso-lines that are continuous, closed and oriented. These properties derive from the fact that the computation of the end point of a segment on the side of a voxel depends only on the values of the vertices of the side, therefore the computation gives the same point for the common side of two adjacent voxels. The label given to that point is opposite for the two voxels, therefore one segment enters the point from one voxel and another segment starts from the point in the other voxel, which gives continuity and orientation. This is also why we give the name of *Marching Lines* to our algorithm. We can march from voxel to voxel in order to follow one bi-iso-line, until the boundary of the image is reached, or the starting point of the line is reached. Hence the reconstructed bi-iso-line is a linked list of points that might be opened or might be closed. The extraction of the lines can be performed by means of a complete scan of the 3D image, or with only selected starting voxels, which can be used to substantially reduce the complexity of the process.

Another way to compute the bi-iso-lines is to consider the iso-surfaces of one of the images, and to label each point of these surfaces using a value interpolated from the second image. The bi-iso-lines are then the lines of the first iso-surfaces that bound regions whose interpolated value in the second image is higher or lower than the second iso-value. If we orient the iso-surfaces of Image 1 so that the normals goes from regions '$-$' toward regions

'+' (see figure 9), then the bi-iso-lines can be oriented with the convention used before, with the left side of the curve bordered by ' +' regions of the second image. If we exchange the two images, the bi-iso-lines have exactly the opposite orientation.

What is remarkable is that the orientation of the reconstructed bi-iso-lines given by our algorithm is compatible with the orientation of the bi-iso-lines defined with the linear interpolation. Specifically, with the convention (summarized in Figure 9), the reconstructed bi-iso-lines correspond to the iso-lines of Image 1 on the iso-surfaces of Image 2. Thus we propose the following heuristic for the arbitrary choice of linking vertices of the 3D polygon P within a given voxel. Link points (from '$-$' to '+'), so that the segments are edges of P. This gives only two solutions, corresponding to two orientations of P. We choose locally between the two, using the sign of the average of the values of Image 1 at the vertices of P. Then the orientation of P is compatible with the orientation conventions expressed in Figure 9.

To conclude, any arbitrary solution which links the vertices of the polygon P can be chosen and leads to continuous closed reconstructed lines. However, only one of these solutions corresponds to the exact topology of the bi-iso-lines as defined in the continuous case with the linear interpolation.

The adaptation of this method to the case of the tetrahedra decomposition is simple. The faces of each tetrahedron must be oriented toward the exterior, in order to label the intersection points on those faces. The polyhedra P_1 and P_2 have at most 4 vertices each, therefore P has at most 4 vertices, and we have only to choose between the two solutions of Figure 6, which can be decided according to the average value of the 4 vertices. Thus the tetrahedra decomposition solution is simpler that the cubic decomposition. A drawback is that the solution can be far from the one given by the tri-linear interpolation (there is no reason that the reconstructed bi-iso-lines have any points in common with the tri-linear solution).

3.2 A solution based on intersection of hyperbolas

This method gives a result which is closer to the tri-linear interpolation solution than the implementations described previously. The method consists in the computation of the intersection, on the sides of the voxels, of the hyperbolas corresponding to the two kinds of iso-surfaces (see Figure 10). The computation is straightforward, because the equation of the hyperbola

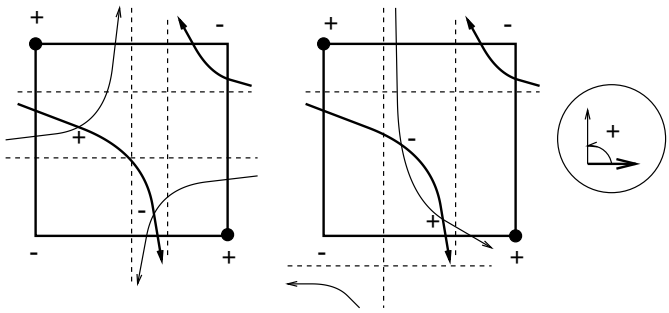

Figure 10: Examples of the intersections of the two hyperboles

resulting from the linear interpolation can be written:

$$\begin{aligned} a_1 + b_1 x + c_1 y + d_1 xy = 0 \\ a_2 + b_2 x + c_2 y + d_2 xy = 0 \end{aligned} \tag{1}$$

The term in xy can be eliminated between the two equations, which gives a linear equation in x and y. Then y can be replaced in one of the two equations, which gives an equation in x of second order. This proves also that there are at most two intersection points on the side of a voxel, except when the branches of the two hyperboles are identical, which constitutes a degenerate case.

As for determining the segments, the intersection points constitute cycles of points (up to 12) around the voxel, which can be labeled '+' or '−', in the same way as described in the previous section (see Figure 10). We could have traced the bi-iso-lines between those points, from points labeled '−' to points labeled '+', using the equations of the two iso-surfaces and with traditional numerical methods (such as the gauss method). In that case, we reconstruct exactly the bi-iso-lines defined with the tri-linear interpolation. But since linear interpolation is already an approximation, and we are nonetheless working with sub-pixel accuracy, we chose to simply link intersection points on voxels with linear segments. Compared with the solutions described previously, the reconstructed bi-iso-lines exactly interpolate the bi-iso-lines defined using tri-linear interpolation at their intersection points on the voxel sides.

4 Crest lines and crest surfaces

The extraction of crest lines from voxel data has been extensively studied by Monga and Benayoun in [11]. They give formulas for computing the curvature of object surfaces based on the first and second derivatives of the image, and give also a method to compute the principal curvatures and principal directions of the 3D surfaces. They define a crest line as a locus of points whose maximal curvature (i.e. maximum absolute value of the two principal curvature), is a local maximum in the corresponding principal direction. They derive also a formula called *extremality*, which involves also the third derivatives of the image, and whose zero crossings correspond to crest lines.

In the following section, we will see that the extremality equation locally defines a surface whose intersection with the iso-surface is the crest line. We will give an independent proof that the formulae derived in [11] corresponds to the curvatures and principal directions of the *iso-surfaces* in the continuous 3D image. We give also new formulae for the curvatures and the principal directions which do not involve the choice of arbitrary vectors in the tangent plane of the iso-surface, and which are symmetric with respect to the principal axes $\vec{x}, \vec{y}$ and $\vec{z}$.

4.1 Curvature and maximal curvature in a 2D image

This example in 2D will help us to understand the computations for the 3D case, which will be developed in the next section. We will see how to compute the curvature of the iso-surface, and derive an equation which gives the points of maximal curvature along the iso-boundary.

Let $f(x, y)$ be the intensity at each point of a smooth continuous bi-dimensional image. The equation of the iso-boundary is $f(x, y) = I$, where I is the fixed iso-value: this is the implicit equation of a curve in the plane. Using the implicit function 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 notation 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:

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

The points of maximum curvature along the iso-boundary satisfy the equation $dc(u)/du = 0$. An equivalent equation is 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 is easily expanded using Equation 5. Equation 6 is of the form $g(x,y) = 0$ and defines a new curve in the 2D image, whose intersection with the iso-boundary $f(x,y) = I$ are the points of maximal curvature. We will call the curve $g(x,y) = 0$ the *maximal curvature curve*. It corresponds to the locus of the maximum curvature points for a continuous variation of the iso-value. What is remarkable is that the maximum curvature curve does not depend on the choice of any iso-value threshold, but is intrinsically defined with the values of the image $f(x,y)$.

The basis of our method is to consider the two images $f(x,y)$ and $g(x,y)$. Then the maximum curvature points of an iso-surface I is the intersection of the iso-boundaries $f(x,y) = I$ of image f and the iso-boundary $g(x,y) = 0$ of image g. Let's see how this method can be extended to 3D images, and how the crest lines can be defined as the intersection of two iso-surfaces.

4.2 The crest surfaces

As with the 2D case, we use the implicit function theorem in order to transform the implicit equation of the iso-surface $f(x,y,z) = I$ of an image

$f(x,y,z)$ 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 [3]) define the principal curvatures and principal directions. These formulae are generally written for the parametric formulation. One of our goals is to find equivalent results with the implicit equation of the surface. The principal curvatures and principal directions correspond to respectively the eigenvalues and the eigenvectors of the matrix:

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

The coefficients are computed as follows. We denote by $\vec{S}(u,v) = (u, v, \phi(u,v))$. Then the parametric definitions yield at a point $\vec{S}(u,v)$ of the surface:

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

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

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:

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

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. However, we will next derive symmetric equations for the curvatures and principal directions. Let k_1, k_2 be the two principal curvatures and $\vec{t_1}, \vec{t_2}$ be the associated principal directions. Then $K = k_1k_2$ is the gaussian curvature and $S = (k_1 + k_2)/2$ is the average curvature of the iso-surface. Then values K and S are also respectively the determinant and half the trace of the matrix given in Equation 7, that is:

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

Using Equation 10:

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

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

where $h = f_x^2 + f_y^2 + f_z^2$. $\vec{x}, \vec{y}, \vec{z}$ have then 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:

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

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 will satisfy the equation:

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

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 coolinear 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 $\vec{t}_{i_2}$ coolinear 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:

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

Some simplifications are possible, we have:

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

Making the substitutions, and then using the linear combination $\vec{t_i} = \frac{f_x - f_z}{f_z}\vec{t_{i_1}} + \frac{f_z - f_y}{f_z}\vec{t_{i_2}}$ restores the symmetry between the three coordinates. We obtain:

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

The vector $\vec{\alpha}$ is more complicated, depending on the first and second derivatives of the image function, but, like $\vec{\beta}$, is symmetric in 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 the $\vec{t_i}$ 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, and for a privileged direction ($\vec{\nabla} f \wedge \vec{w} = 0$), which is troublesome.

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

If k_1 is the maximum curvature ($|k_1| \geq |k_2|$), then, the gradient of k_1 in the direction of $\vec{t_2}$ is given by $\vec{\nabla} k_1 \cdot \vec{t_2}$, which can be computed from Equations 14 and 18. Note however that the eigenvalue k_1 is associated with the eigenvector $\vec{t_1}$, but k_1 is the curvature *about* the axis $\vec{t_1}$ and therefore *in the direction of* $\vec{t_2}$). Setting the directional derivative to zero, we obtain a *"crest surface"* $g(x, y, z) = 0$, containing points where the value of k_1 is extremal in the $\vec{t_2}$ direction. The intersection of the crest surface with the iso-surface of the image gives the crest lines. The Marching Lines algorithm

can then be used to 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 g is not defined. In that case, one simple solution is to stop the "marching" when g is not defined for one vertex of the voxel. 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-contour belongs to the boundary of an object) and the absolute value of the curvature. The exact formula and thresholds used to post-filter the crest lines computed with the Marching Lines algorithm will depend on the particular application, and is not of interest here. The important point is that we have reduced the study of a 3D image to a 1D variety of points: the bi-iso-lines defined with the intersection of the iso-surfaces $f(x, y, z) = I$ and $g(x, y, z) = 0$. In the case of the results presented in this paper, we have simply discarded the lines whose length was inferior to a given number of nodes, which proved to be a convenient way to keep only the most significant crest lines.

4.3 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 (for the same reason as a result of geometry, sometimes refered to as the "hairy ape" theorem, which states that a furry sphere cannot be combed without creating a cowlick). Also, because of the sign of the extremality function $g(x, y, z)$ is meaningless, the crest surface separates regions with '+' and '$-$' extremality, and sometimes those regions can switch in the way of a chess board. This can be seen in the example of Figure 11, for 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 compute locally the orientation of the principal directions. More precisely, when the crest line enters into a new voxel, the values for the principal directions and

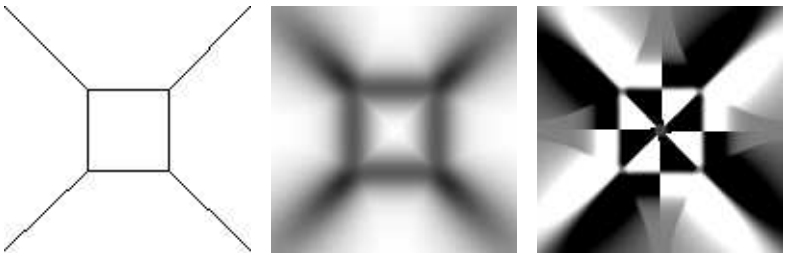

Figure 11: Crest lines, maximum curvature and extremality for $z = x^4 + y^4$

extremality of the entering face are stored at each vertex of the face, and only the values at the four other vertices of the cube must be computed. We will call the *maximal direction* the principal direction $\vec{t_m}$ whose eigenvalue k_m is the maximum curvature. The direction $\vec{t_m}$ is approximately tangent to the crest line. The other principal direction $\vec{t_p}$ is computed as the cross product of $\vec{t_m}$ with the gradient direction: $\vec{t_p} = \frac{\vec{t_m} \wedge \vec{\nabla} f}{||\vec{\nabla} f||}$.

At each of the four vertices on the opposite face of the entrance face, a choice must be made as to the orientation of $\vec{t_p}$ and $\vec{t_m}$. The choice will affect the sign of the extremality function at each vertex, and thus influence from which face the crest line exits. We determine the maximum direction $\vec{t_{m_{in}}}$ at point of entrance of the crest line, using linear interpolation of the direction at the four vertices of the entering face. We choose orientations of the maximum direction $\vec{t_m}$ at the other four vertices on the opposite face so that the scalar product with $\vec{t_{m_{in}}}$ is positive. Implicitly, we are assuming that the crest line direction does not change of more than 90 degrees within a voxel. The orientation of $\vec{t_m}$ defines the orientation of $\vec{t_p}$ which in turn determines the sign of the extremality function. The point where the crest line leaves the cube is computed with the Marching Lines algorithm. As we can see, the sign of the values at the vertices of the voxel is not deterministic because it depends on the direction of the entering crest line, but the continuity of the crest line is still guaranteed. The Marching Lines algorithm as modified is given in Figure 12.

This algorithm requires that previously computed segments are stored and retrieved, which can be done efficiently with a hash table.

for each voxel in the image
    if this voxel belongs to the iso-surface
        compute the extremality for the 8 vertices
        for each intersection point on a face of the voxel {
            give an arbitrary orientation to the crest line
            use this orientation to recompute the extremality
            follow the crest line into the 3D image
            until the edge of the image is reached
            or the extremality is not defined
            or an already computed crest line is reached }

Figure 12: Modification of the Marching Lines algorithm

5 Applications

Many applications can be derived from the two algorithms described previously; the Marching Lines algorithm and the computation of iso-surface characteristics such as the gaussian and mean curvatures, principal curvatures, principal directions, and "extremality". The Marching Lines algorithm can be used to extract any characteristic line of the 3D image defined by means of the surface characteristics computed with the second algorithm. Figure 13 shows the iso-surface corresponding to the bones in the 3D CT scan of a head, rendered with a conventional lighting model. Figure 14 shows the same iso-surface, but this time, the color corresponds to the value of the maximum curvature (with black points corresponding to high curvature points). Figure 15 is the result of the Marching Lines algorithm applied to the image of the head (64^3 pixels), without any post-filtering of the computed 3D lines. It shows the subpixel accuracy of the solution. However, it is difficult to visualize in a single view the quality of the result. The quality of the result is much more apparent when the mesh of 3D lines is rotating, or with a stereoscopic pair of images.

Figure 16 also demonstrates the application of the Marching Lines algorithm to two different 3D images of the same subject, taken in two different positions. The 3D lines have been filtered to retain only those lines whose

numbers of points is greater than twenty, to keep only the most significant lines. Once again, only stereoscopic techniques can be used to fully appreciate the result. Particularly, there are internal features of the skull that correspond to very significant crest lines, and that are overlaid in this single view with more significant crest lines, such as the orbital lines, or the sub-mandibular line.

Figure 17 is the result of the automatic 3D line registration, based on geometric hashing, as described in [5], which takes a few seconds of cpu time for the example of the skull. At the end of the process, we have the exact 3D transform between the two 3D data sets. We have implemented the Marching Lines algorithm with random starting points scattered among the image to compute the crest lines. The computation time, including the computation of the differentials of the image with linear filtering requested to compute the extremality function, is about 1 minute for each of the 3D image. Because crest lines are 1D varieties of points, we have verified experimentally that the complexity of our method is sub-linear with respect to the total number of voxels (cpu time is only increased by a factor of two if the number of voxels is multiplied by eight). The bottleneck of the method is the application of a smoothing filter to the whole image, prior to the extraction of the 3D lines, which takes about one half hour (all cpu times are given for a DEC 5000 workstation), and whose complexity is linear in the number of voxels (see [12]).

We can also compute other characteristic lines from the voxel values, such as the *"iso-gaussian"* lines, which are lines that separate regions of the iso-surface whose gaussian curvature k is higher or lower than a given threshold (a threshold of zero gives the parabolic lines). Unlike the extremality function, k is given by Equation 12, which is global. Therefore, the iso-gaussian lines can be computed without marching from voxel to voxel, but directly with a local computation of the segments in each voxel. Figure 18 are the parabolic lines of the iso-surface of a synthetic torus, extracted with the Marching Lines algorithm.

6 Conclusion

To conclude, we have described a very powerful and general-purpose tool for line extraction from 3D images, called the Marching Lines algorithm.

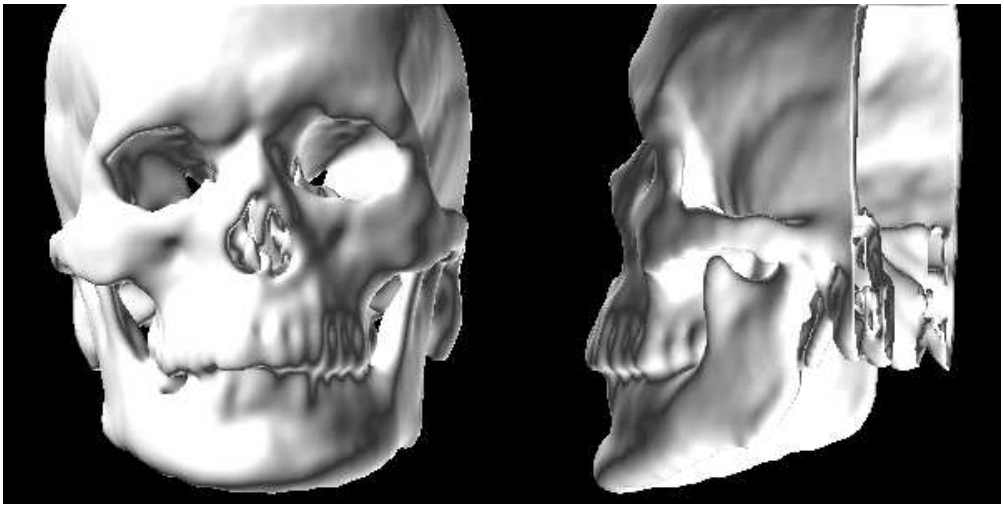

Figure 13: Iso-surface of a 3D scanner image

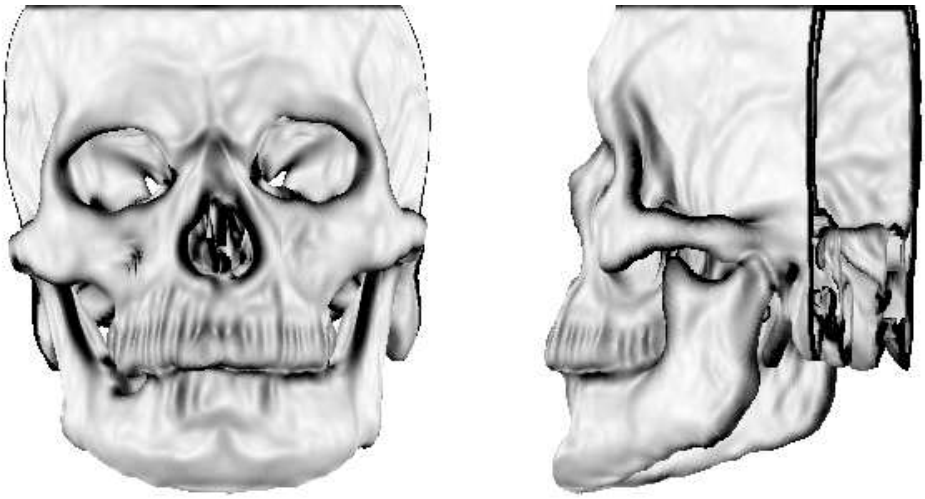

Figure 14: Computation of the maximum curvature

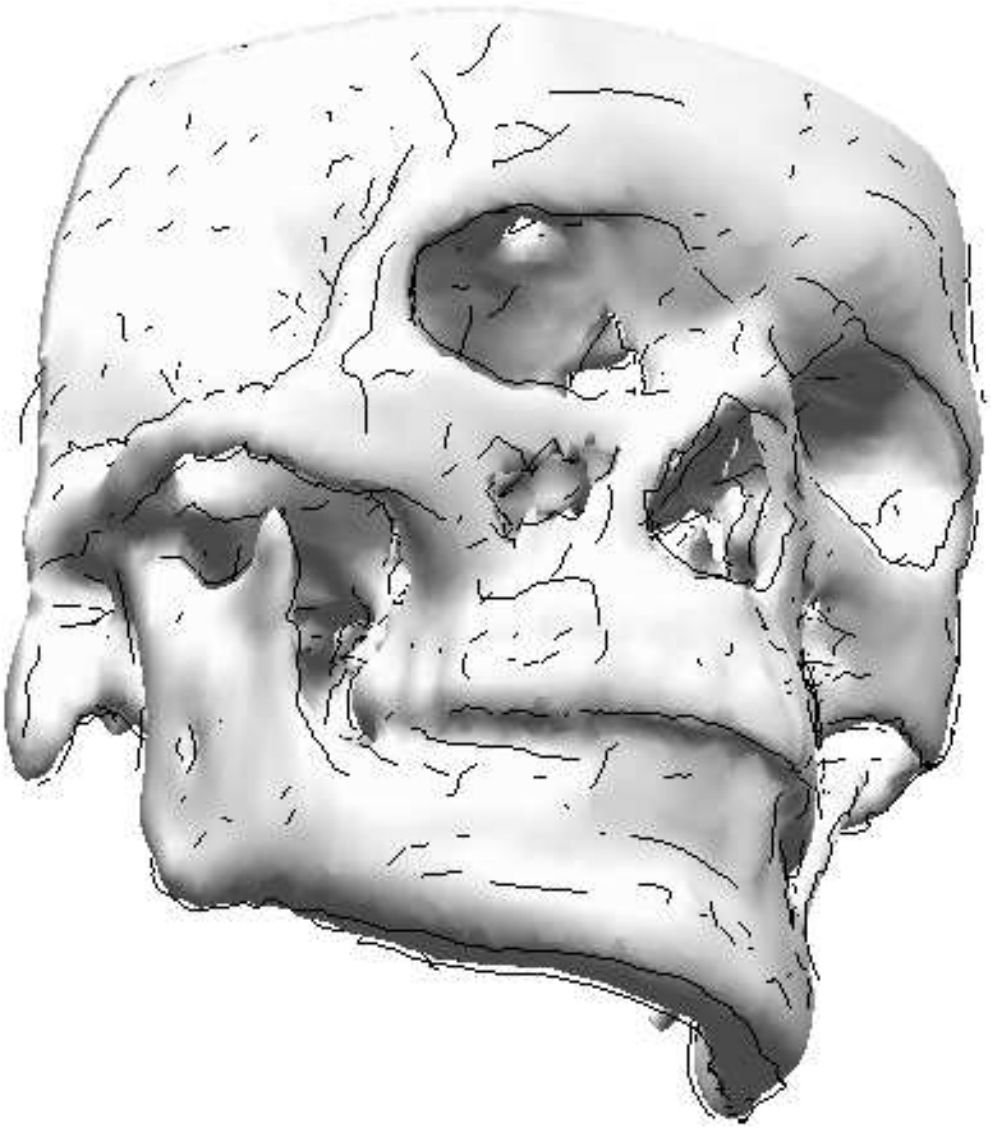

Figure 15: Crest lines without filtering, superimposed on the iso-surface

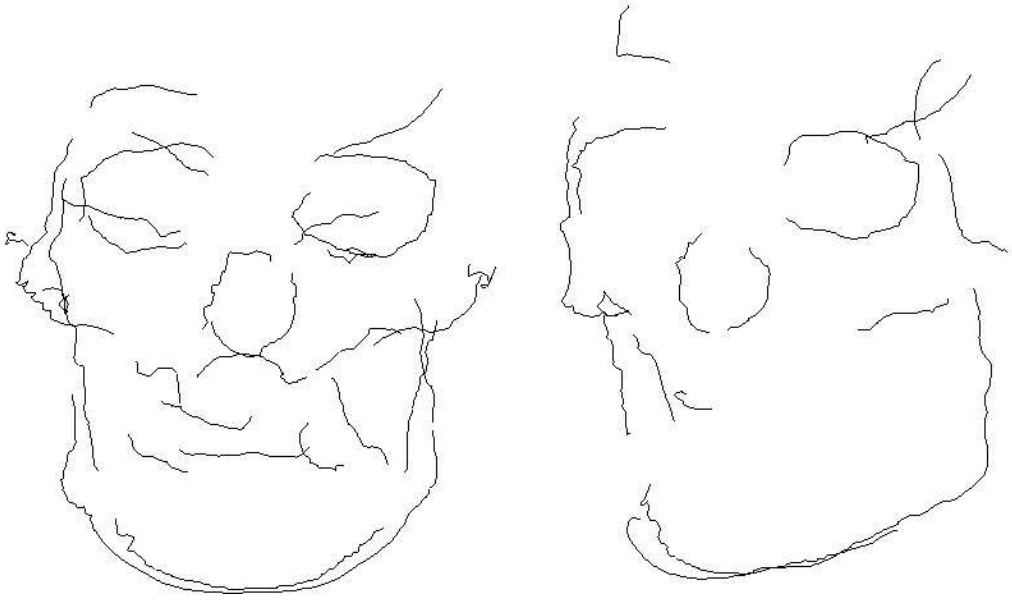

Figure 16: Filtered crest lines for two 3D data sets

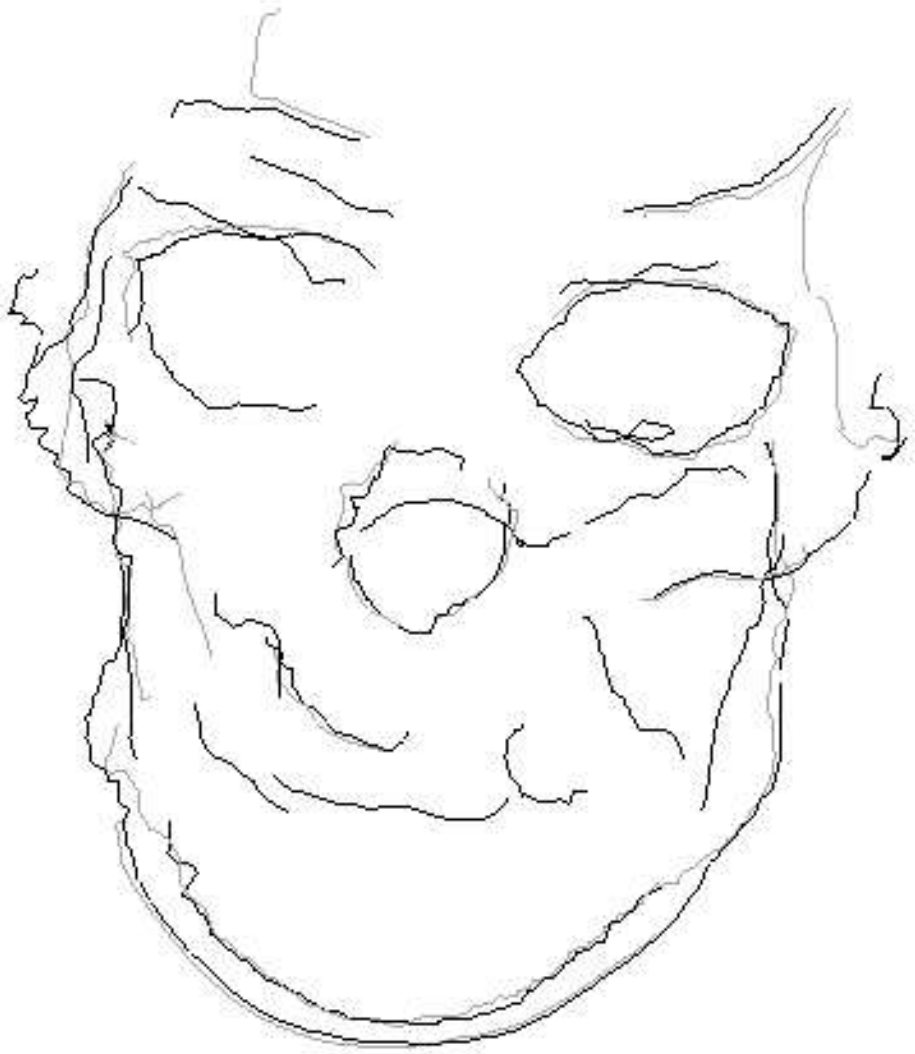

Figure 17: 3D registration of the two crest line sets (black and grey)

Figure 18: Parabolic lines in the synthetic 3D image of a torus

It is powerful because it guarantees good topological properties of the reconstructed lines, and general-purpose because it only requires as input two images and two iso-values. We have presented in the second part of this paper a new method to compute the differential features of the iso-surfaces, and we have shown how to use the Marching Lines algorithm to extract characteristic lines, such as crest lines and iso-gaussian curves. 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. 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 the paper.

References

[1] John Canny. A computational approach to edge detection. *IEEE PAMI*, 8(6):679–698, November 1986.

[2] Rachid Deriche. Using canny's criteria to derive a recursively implemented optimal edge detector. *International Journal of Computer Vision*, 6:167–187, November 1986.

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

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

[5] 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.

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

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

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

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

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

[11] O. Monga and S. Benayoun. Using partial derivatives of 3d images to extract typical surface features. *rapport de recherche INRIA*, March 1992.

[12] O. Monga and R. Deriche. 3d edge detection using recursive filtering. *IEEE Conference on Vision and Patern Recognition*, June 1989.

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

[14] Jane Wilhelms and Allen Van Gelder. Topological ambiguities in iso-surface 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.

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