Crest lines extraction in volume 3D medical images : a multi-scale approach

Olivier Monga, Richard Lengagne, Rachid Deriche

1994

ISSN 0249-6399

ROCQUENCOURT

Crest lines extraction in volume 3D medical images : a multi-scale approach

Olivier MONGA, Richard LENGAGNE, Rachid DERICHE

Programme 4 — Robotique, image et vision

Projet SYNTIM

Rapport de recherche n ° 2338 — Juillet 1994 — 34 pages

Abstract: Recently, we have shown that the differential properties of the surfaces represented by 3D volumic images can be recovered using their partial derivatives. For instance, the crest lines can be characterized by the first, second and third partial derivatives of the grey level function $I(x, y, z)$. In this paper, we show that :

- the computation of the partial derivatives of an image can be improved using recursive filters which approximate the Gaussian filter,
- a multi-scale approach solves many of the instability problems arising from the computation of the partial derivatives,
- we illustrate the previous point for the crest line extraction (a crest point is a zero-crossing of the derivative of the maximum curvature along the maximum curvature direction).

We present experimental results of crest point extraction on synthetic and 3-D medical data.

Key-words: Volume 3D medical images, surface modelling, curvatures, crest lines, multi-scale derivation, recursive filtering

(Résumé : tsvp)

Extraction de lignes de crêtes dans des images volumiques 3D médicales : une approche multi-échelle

Résumé : Récemment, Nous avons montré que les propriétés différentielles des surfaces peuvent être calculées à partir des dérivées partielles des images volumiques 3D. Par exemple, les lignes de crête peuvent être caractérisées à l'aide des premières, deuxièmes et troisièmes dérivées partielles de la fonction des niveaux de gris $I(x, y, z)$. Dans cet article nous montrons que :

- le calcul des dérivées parielles d'une image peut être amélioré en utilisant des filtres récursifs approximant le filtre gaussien et ses dérivées,
- une approche multi-échelle résoud un bon nombre des problèmes d'instabilité liés au calcul des dérivées partielles,
- nous illustrons le précédent point pour l'extraction des lignes de crête (un point de crête est un passage par zéro de la dérivée de la courbure maximum le long de la direction maximum de courbure).

Nous présentons des résultats expérimentaux sur l'extraction de lignes de crêtes sur des données synthétiques et sur des données médicales réelles.

Mots-clé : Images volumiques 3D médicales, modélisation de surfaces, courbures, lignes de crêtes, dérivation multi-échelle, filtrage récursif

1 Introduction

Volumic 3D images are now widely distributed in the medical field [8, 20, 7, 1]. They are produced from various modalities such as Magnetic Resonance Imagery (MRI), Computed Tomography Imagery (CT), Nuclear Medicine Imagery (NMI) or Ultrasound Imagery (USI). Such data are represented by a discrete 3D grey level function $I(i, j, k)$ where the high-contrast points (3D edge points) correspond to the discrete trace of the surfaces of the geometrical structures [15, 14, 23]. A motivating issue is then to extract typical features of these surfaces. The most natural way is to look for differential Euclidean surface invariants such as : curvatures, crest lines, parabolic lines, umbilic points... [10, 18, 17], [19, 9, 13, 2, 21, 16]. Recently, we have shown that the differential properties of a surface defined by an iso-contour in a 3D image can be recovered from the partial derivatives of the corresponding grey level function [13]. In [13] crest lines are extracted using first, second and third order partial derivatives provided by 3D Deriche filters [14, 15]. The critical point of this approach also studied in [21] is the stability of expressions including second and third order partial derivatives such as the "extremality criterion" defined in [13, 21].

In this paper we propose recursive 3D filters to improve the computation of partial derivatives and also a multi-scale approach to extract the zero-crossings of the extremality criterion.

In Section II, we show that derivative filters coming from isotropic (rotation invariant) smoothing filters should be used to ensure the Euclidean invariance of the curvatures. Then we derive an algorithm to compute first, second and third order partial derivatives of a 3D volumic image. These derivatives are used to obtain curvatures invariant by rigid motion (Euclidean invariant).

Section III deals with the computation of the curvatures of the surfaces traced by the iso-contours (3D edge points) from the partial derivatives of the image (for instance provided by the previous method). This section recalls the main results of the reference [13] and shows the problems induced by a single scale filtering.

In Section IV, we propose to use different widths of filters to compute the curvatures. This leads to a multi-scale curvature computation scheme where the scale is the width

of the filters. We apply this principle to track the zero-crossings of the derivative of the maximum curvature points along the maximum curvature direction (extremality criterion) which correspond to the crest points. The zero-crossings coming from the different scales are merged using a valuated adjacency graph. We propose some simple and efficient strategies to extract stable zero-crossings from this graph.

In Section V we present experimental results obtained on synthetic and real data (CT and MR 3D images). We show that our approach combining a multi-scale scheme and also the use of better filters provides reliable crest lines even for noisy data.

2 Computation of the partial derivatives of a 3D image using linear filters

2.1 Introduction

Recently R. Deriche has introduced recursive filters to approximate the Gaussian filter and its derivative [5]. First of all, we show the advantage of using such filters to compute differential Euclidean invariants. We recall the main results reported in [5]. Then, we extend Deriche's work to 3D and to the computation of third order derivatives. We also show how to normalize these filters in order to obtain coherent values for first, second and third order derivatives. We develop an efficient algorithm to compute the partial derivatives of a 3D image.

2.2 Statement of the problem

In this section the theoretical properties of the filters will be discussed in the continuous space (an image is $I(x,y,z)$) and the implementation in the discrete space (an image is $I(i,j,k)$)

Let $I(x,y,z)$ be a 3D image.

We are looking for the partial derivatives of $I(x,y,z)$: $\dfrac{\partial^n(I(x,y,z))}{\partial x^m \partial y^p \partial z^q}$, $n = m+p+q$ that we represent using the subscript notation : $I_{x^m y^p z^q}$ (we will write only the variables, the power of which is not zero, for instance $I_{x^1 y^0 z^0}$ becomes I_x).

If $f(x,y,z)$ is the impulse response of a smoothing filter, the restored image I_r is equal to $I * f$, where $*$ is the convolution product. Classically, using the properties of the convolution product we obtain

$$\frac{\partial^n I_r}{\partial x^m \partial y^p \partial z^q} = \frac{\partial^n (I * f)}{\partial x^m \partial y^p \partial z^q} = I * (\frac{\partial^n f}{\partial x^m \partial y^p \partial z^q})$$

Then the impulse response of the filter which computes $I_{x^m y^p z^q}$ is $\frac{\partial^n f}{\partial x^m \partial y^p \partial z^q}$.

We develop a popular scheme which reduces the search of derivative filters of any order to the search of a smoothing filter [3, 5, 4, 6]. The question is now : what are the suitable properties for our smoothing impulse response if we are interested in the computation of Euclidean differential invariants ?

2.3 Isotropy of the filters and invariance properties

In the reference [13] we set the smoothing operator to :

$$f(x,y,z) = f_0(x) f_0(y) f_0(z)$$

with

$$f_0(x) = c_0(1 + \alpha \mid x \mid)e^{-\alpha|x|}$$

where c_0 is a normalization constant ; f_0 is Deriche's 1D smoothing operator [6].

The advantage of using this function is that we obtain recursive filters which are optimal for Canny's criteria in the direction of the frame axis X, Y, Z. The drawback is that our 3D smoothing operator $f(x,y,z)$ is not isotropic i.e. not rotationally invariant. For instance, this implies that the magnitude of the gradient computed with the corresponding derivative filters is not invariant by a rigid motion. In the references [10, 11, 14] we carefully illustrate this point by showing that the magnitude of a 3D edge depends on its orientation. Similarly, we could also show this for the Laplacian and any other Euclidean differential invariants (e.g. curvatures).

We now show that if we use partial derivatives implemented by filters derived from an isotropic impulse response then the differential invariants remain invariant.

Let $I(x,y,z)$ be a 3D image.

Let $J(x, y, z)$ be the image I transformed by a rigid motion :

$$J(X_0, Y_0, Z_0) = I(x_0, y_0, z_0)$$

with

$$(X_0, Y_0, Z_0)^t = \mathbf{R}(x_0, y_0, z_0)^t + \mathbf{T}$$

where $\mathbf{R}$ and $\mathbf{T}$ are respectively the matrices defining a rotation around the origin and a translation.

Let f be the impulse response of an isotropic smoothing filter, we have :

$$f(x_1, y_1, z_1) = f(x_2, y_2, z_2)$$

where

$$(x_2, y_2, z_2)^t = \mathbf{R}(x_1, y_1, z_1)^t$$

thus :

$$(I * f)(x_0, y_0, z_0) = (J * f)(X_0, Y_0, Z_0)$$

Let E be a function of the partial derivatives of I. If E is an Euclidean differential invariant of I we have :

$$E((I_x)(x_0, y_0, z_0), (I_y)(x_0, y_0, z_0), (I_z)(x_0, y_0, z_0), (I_{xx})(x_0, y_0, z_0)), (I_{xy})(x_0, y_0, z_0)...) =$$
$$E((J_x)(X_0, Y_0, Z_0), (J_y)(X_0, Y_0, Z_0), (J_z)(X_0, Y_0, Z_0), (J_{xx})(X_0, Y_0, Z_0)), (J_{xy})(X_0, Y_0, Z_0)...)$$

If we compute the partial derivatives using our filters we obtain :

$$E((I * f_x)(x_0, y_0, z_0), (I * f_y)(x_0, y_0, z_0), (I * f_z)(x_0, y_0, z_0)....) =$$
$$E((I * f)_x(x_0, y_0, z_0), (I * f)_y(x_0, y_0, z_0), (I * f)_z(x_0, y_0, z_0)...)$$

Given that E is also an Euclidean differential invariant of $I * f$, the last expression is equal to :

$$E((J * f)_x(X_0, Y_0, Z_0), (J * f)_y(X_0, Y_0, Z_0), (J * f)_z(X_0, Y_0, Z_0)...) =$$
$$E((J * f_x)(X_0, Y_0, Z_0), (J * f_y)(X_0, Y_0, Z_0), (J * f_z)(X_0, Y_0, Z_0)...)$$

Therefore we have shown that :

$$E((I * f_x)(x_0, y_0, z_0), (I * f_y)(x_0, y_0, z_0), (I * f_z)(x_0, y_0, z_0)....) =$$
$$E((J * f_x)(X_0, Y_0, Z_0), (J * f_y)(X_0, Y_0, Z_0), (J * f_z)(X_0, Y_0, Z_0)...)$$

The previous equation means that E is also an Euclidean invariant if the partial derivatives are provided by the filters defined by f.

We point out that there are two types of differential Euclidean invariants :

- the differential Euclidean invariants for any point of the image : gradient, laplacian, curvatures of the surface (hypersurface in the 3D case) defined by the image [12]
- the differential Euclidean invariants defined only at the edge points ; we assume that the iso-contours define the trace of a curve (surface in the 3D case) such as at each point of this surface the gradient corresponds to the normal ; in this specific case, the curvatures of the curve (surface) can be expressed using the partial derivatives of the image ; the corresponding expressions are therefore Euclidean invariant only at the edge points.

The preceding proof is also valid in the second case by assuming that the gradient is parallel to the normal for $I * f$ which is a reasonable assumption.

Therefore, we have shown that if the partial derivatives of an image are computed using filters, derived from an isotropic smoothing filter then the differential Euclidean invariants (calculated using these partial derivatives) are invariants by a rigid motion (Euclidean invariants).

Figure 1 shows the interest of using an isotropic filter when computing the partial derivatives; we compare the curvatures at the edges of a disk (a circle). The derivatives are successively computed with the anisotropic Deriche filter and with the recursively implemented approximation of the derivatives of isotropic Gaussian filter. We choose the width of the filters such that the curvature (inverse of the radius of the circle) has the right value (in pixels). We see clearly the false variations of the curvature induced by filters derived from an anisotropic smoothing filter.

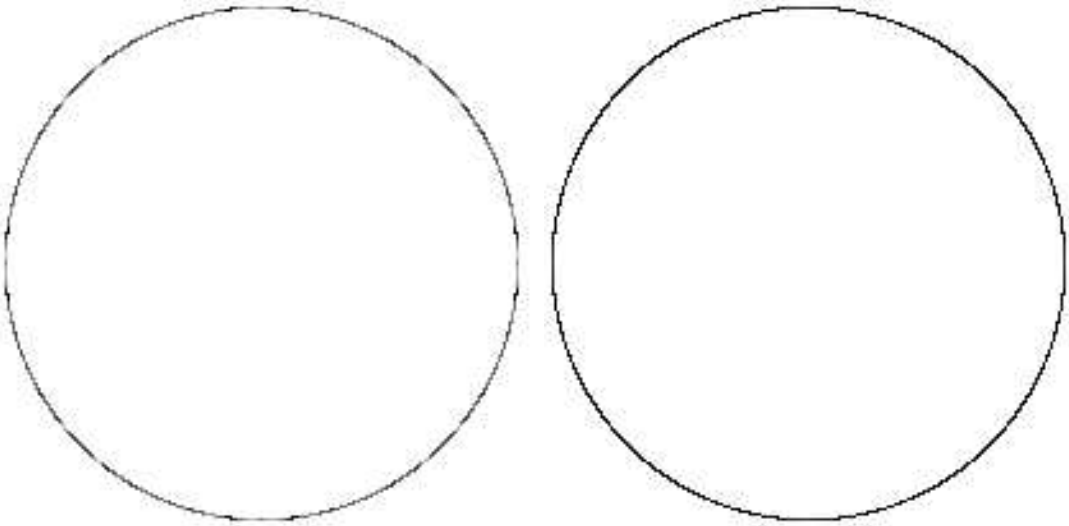

Figure 1: Right image : Curvature values using filters derived from an anisotropic smoothing filter (Deriche filter) ; Left image : Curvature values using filters derived from an isotropic filter (Gaussian filter)

2.4 Recursive filters to approximate Gaussian filters

2.4.1 Introduction

The result of the previous section clearly shows the great interest of computing the partial derivatives of an image using filters derived from an isotropic smoothing impulse response. Otherwise we can obtain gradient [14], Laplacian or curvatures [13] which are not invariant by a rigid motion. On the other hand, we also take interest in using separable recursive filters in order to obtain a reasonable computational cost. A way to join these two antagonist points is to use the recursive approximation of the Gaussian filter (the only separable non trivial smoothing filter) introduced by R. Deriche in the recent reference [5].

2.4.2 Recursive isotropic filters for first and second derivatives in the 2D case

Here, we recall the main result of the reference [5]. The 1D Gaussian smoothing filter is given by :

$$g(x) = e^{-\frac{x^2}{2\sigma^2}}$$

Using Prony's method, the positive and negative part of g and of its normalized derivatives of first and second order can be approximated (with a normalized square error of about $\epsilon^2 = 10^{-6}$) by a 4th order Recursive operator (IIR)(see Appendix A) :

$$h(x) = (a_0 \cos(\frac{\omega_0}{\sigma}x) + a_1 \sin(\frac{\omega_0}{\sigma}x))e^{-\frac{b_0}{\sigma}x} + (c_0 \cos(\frac{\omega_1}{\sigma}x) + c_1 \sin(\frac{\omega_1}{\sigma}x))e^{-\frac{b_1}{\sigma}x} \qquad (1)$$

Therefore, the 1D Gaussian filter and its first and second order derivatives can be recursively implemented. Using the separability property, we derive directly the recursive filters to compute the first and second order derivatives of a 2D image.

2.4.3 Extension of Deriche Gaussian filters to third order derivatives and to the 3D case

Using the separability of the filters we extend directly the previous scheme to the 3D case. Following the notations of Section II.3 this amounts to setting :

$$f(x, y, z) = g(x)g(y)g(z)$$

We also extend this filtering scheme to the third order derivative case. We develop a set of recursive filters approximating the Gaussian and its derivatives which can be used to compute the first, second, and third order derivatives of a 3D image.

We stress that a very important point not carried out in [5] is the normalization of the filters which allows to obtain coherent values for the different derivatives. Here we use the scheme presented in [13] :

Let f_0, f_1, f_2, f_3 be the convolution filters to provide respectively the smoothed zero, first, second, and third derivatives of a function of a single variable. The normalization coefficients are chosen so that :

$$\int_{-\infty}^{+\infty} f_0(x)dx = 1;$$

$$\int_{-\infty}^{+\infty} x f_1(x)dx = 1;$$

$$\int_{-\infty}^{+\infty} f_2(x)dx = 0 \text{ and } \int_{-\infty}^{+\infty} \frac{x^2}{2} f_2(x)dx = 1;$$

$$\int_{-\infty}^{+\infty} \frac{x^3}{6} f_3(x)dx = 1 \text{ and } \int_{-\infty}^{+\infty} x f_3(x)dx = 0.$$

These conditions ensure that convolution by the filters f_0, f_1, f_2, f_3 yield the correct derivatives when applied to polynomials. When the filters are to be applied in a discrete setting, a similar scheme is used to find the normalization coefficients.

In the appendix A, we present the details of the filters and of their recursive realization.

2.5 Algorithm to compute the first, second and third derivatives of a 3D volumic image

Let $f_0(x)$, $f_1(x)$, $f_2(x)$ and $f_3(x)$ be respectively the smoothing, first derivative, second derivative and third derivative impulse responses of the filters defined in the previous section. These filters can be implemented recursively and approximate with a very low error (see appendix) the Gaussian filter and its first, second and third derivatives. We extend to second and third order derivatives the 3D recursive filtering sheme described in [14]. We obtain for the computation of $\frac{\partial^n f}{\partial x^m \partial y^p \partial z^q}$, $m+p+q=3$ the following algorithm where the convolution products are implemented using the recursive implementation of the filters :

$$\text{for } (m,p,q)/(m+p+q) \leq 3 \text{ do}$$
$$\left[\begin{array}{l} R = I \\ R = R * f_m(x) \\ R = R * f_p(y) \\ I_{x^m y^p z^q} = R * f_q(z) \end{array}\right.$$

3 Using the partial derivatives of a 3D volumic image to extract and to characterize 3D structures

3.1 Introduction

In this section we show how to use the partial derivatives computed by the filters proposed in the previous section.

3.2 3D edge detection

Classically, 3D edge detection can be done by computing the first derivatives (gradient approach) or the Laplacian (Laplacian approach) of the image [23]. Recently, recursive filtering has been introduced to define 3D edge detection operators having a better noise immunity and a lower computational cost [15, 14]. The drawback of the recursive filters proposed in [15, 14] is that their output depends on the orientation of the edge.

The use of the recursive filters presented in the previous section enables us to avoid this drawback. We thus put them instead of Deriche filters in the recursive filtering scheme described in [14].

3.3 Curvatures from partial derivatives

Here we recall the main results of the reference [13]

3.3.1 2D case

Let (C) be a contour in an image $I(x, y)$. If we assume that the gradient $\vec{g}$ at a point of (C)

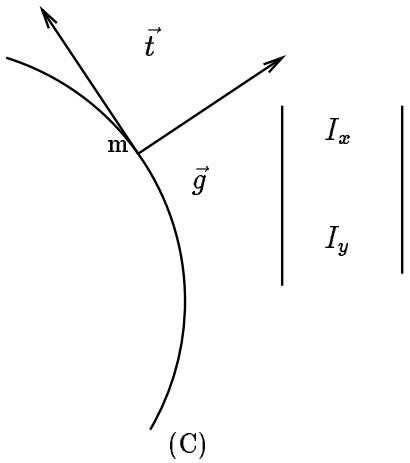

Figure 2: $2D$ edge

is parallel to the normal to (C) (this means that the edge is at least locally an iso-contour), the curvature k at point m is [13] :

$$k = \frac{-t^T H t}{\|\vec{g}\|}$$

where H is the Hessian of the grey level function $I(x, y)$:

$$H = \begin{bmatrix} I_{xx} & I_{xy} \\ I_{xy} & I_{yy} \end{bmatrix}$$

and $\vec{t}$ is the curve unit tangent vector to (C) at point m :

$$\vec{t} = \frac{(-I_y, I_x)^t}{\sqrt{I_x^2 + I_y^2}}$$

3.3.2 3D case

Let (C) be a 3D contour defining a surface $(\sum)$ in an image $I(x, y, z)$. The curvature k_t in a

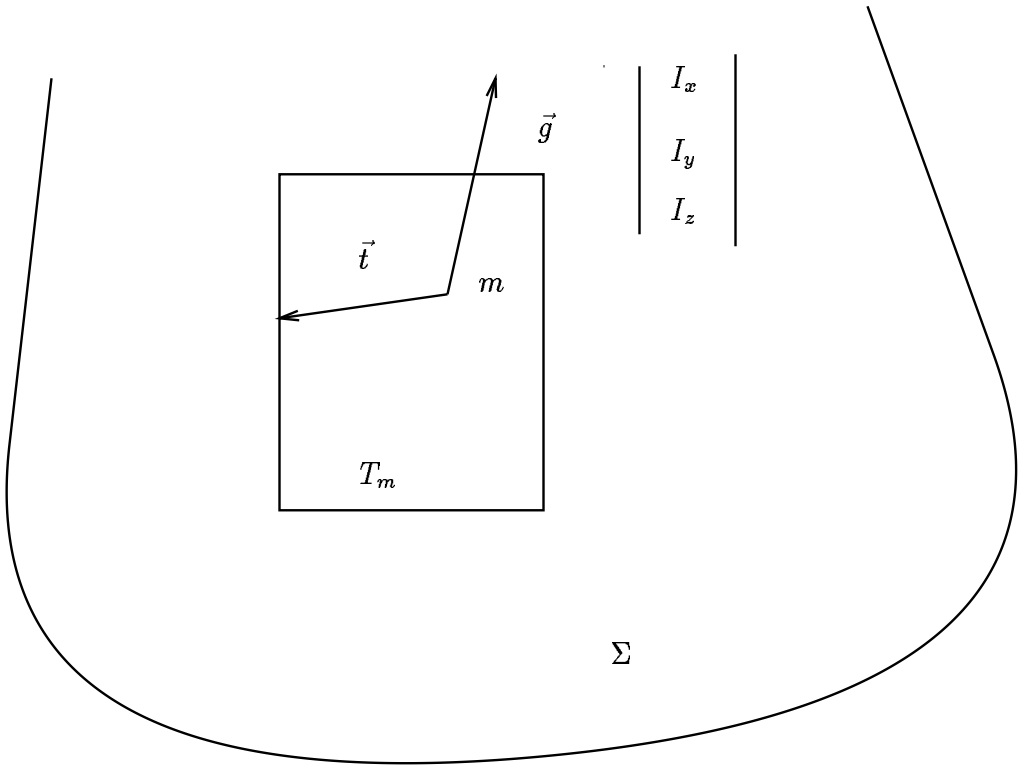

Figure 3: $3D$ edge

given direction $\vec{t}$ is given by the formula:

$$k_t = \frac{-t^T H t}{||\vec{g}||}$$

If we assume that the gradient $\vec{g}$ at a point of $(\sum)$ is parallel to the normal to $(\sum)$ (this means that the edge is at least locally an iso-contour), the two principal curvatures K_i and two principal curvature directions d_i are :

$$K_i = \frac{h^T H h + f^T H f \pm \sqrt{(h^T H h - f^T H f)^2 + 4(h^T H f)^2}}{2 \ \|\vec{g}\|} \tag{2}$$

$$d_i \ = \begin{pmatrix} h_1 + f_1 \frac{\|\vec{g}\| K_i - h^T H h}{f^T H h} \\ h_2 + f_2 \frac{\|\vec{g}\| K_i - h^T H h}{f^T H h} \\ h_3 + f_3 \frac{\|\vec{g}\| K_i - h^T H h}{f^T H h} \end{pmatrix} \text{ with } i = 1, 2 \tag{3}$$

where

$$\begin{pmatrix} \vec{g} & \vec{h} & \vec{f} \end{pmatrix} = \begin{pmatrix} I_x & \frac{I_y}{\gamma} & \frac{I_z I_x}{\gamma \|\vec{g}\|} \\ I_y & \frac{-I_x}{\gamma} & \frac{I_y I_z}{\gamma \|\vec{g}\|} \\ I_z & 0 & -\frac{\gamma}{\|\vec{g}\|} \end{pmatrix}$$

with $\gamma = \sqrt{I_x^2 + I_y^2}$

and $H = \begin{bmatrix} I_{xx} & I_{xy} & I_{xz} \\ I_{xy} & I_{yy} & I_{yz} \\ I_{xz} & I_{yz} & I_{zz} \end{bmatrix}$

3.4 Ridge points from partial derivatives

3.4.1 2D case

The curvature extrema are located where the derivative $k'(s)$ of the curvature $k(s)$ with respect to the arc length s is zero i.e. :

$$\frac{\frac{d(t^T H t)}{ds} \|\vec{g}\| \ - \ (t^T H t) \ \frac{d\|\vec{g}\|}{ds}}{\|\vec{g}\|^2} \ = \ 0$$

We obtain finally the following equation :

$$\frac{3 \ (t^T H t)(g^T H t) \ - \ \|\vec{g}\|^2 \ t^T \begin{pmatrix} t^T H_x t \\ t^T H_y t \end{pmatrix}}{\|\vec{g}\|^3} \ = \ 0$$

where

$$H_x = \begin{bmatrix} I_{xxx} & I_{xxy} \\ I_{xxy} & I_{xyy} \end{bmatrix}$$

and

$$H_y = \begin{bmatrix} I_{xxy} & I_{xyy} \\ I_{xyy} & I_{yyy} \end{bmatrix}$$

3.4.2 3D case

Let $\vec{d_1}$ be the maximum curvature direction, the derivative of the maximum curvature K_1 in the direction $\vec{d_1}$ is denoted as $\nabla_{\vec{d_1}} K_1$:

$$\nabla_{\vec{d_1}} K_1(m) = \nabla\vec{K}_1.\vec{d_1} = \frac{\|\vec{g}\|^2 d_1^T \cdot \begin{bmatrix} d_1^T H_x d_1 \\ d_1^T H_y d_1 \\ d_1^T H_z d_1 \end{bmatrix} - \left(d_1^T H d_1)(d_1^T H g\right)}{\|\vec{g}\|^3}$$

with H being the Hessian of I and H_x, H_y, H_z its partial derivatives. In the sequel we will call $\nabla_{\vec{t_1}} K_{t_1}(m)$ the "extremality criterion".

We can also compute the derivative of the minimum curvature K_2 in the minimum curvature direction $\vec{d_2}$, i.e. :

$$\nabla_{\vec{d_2}} K_2(m) = \nabla\vec{K}_2.\vec{d_2} = \frac{\|\vec{g}\|^2 d_2^T \cdot \begin{bmatrix} d_2^T H_x d_2 \\ d_2^T H_y d_2 \\ d_2^T H_z d_2 \end{bmatrix} - \left(d_2^T H d_2)(d_2^T H g\right)}{\|\vec{g}\|^3}$$

in order to extract the minima and the maxima of the minimum curvature in the minimum curvature direction.

3.5 Practical computation of the crest lines of the surfaces in a 3D image at a given resolution

The main stages of our algorithm allowing to extract crest lines in a 3D image are :

1. Computation of the first, second and third order partial derivatives of the image $I(x,y,z)$ ($\frac{\partial^n f}{\partial x^m \partial y^p \partial z^q}$, $m+p+q=3$) using the recursive filters defined in Section 2 for a given value of σ ;

2. Extraction of the 3D edge points using the first order partial derivatives (gradient) of I (see section 3.2) ;

3. For each point of the 3D edge map, computation of :

   - the two principal curvatures and the corresponding principal curvature directions using the formulas of Section III.3 ;
   - the extremality criterion (derivative of the maximum curvature along the corresponding principal direction) using the formulas of Section III.4.

4. Building of a extremality criterion image $C_\sigma(x, y, z)$ such as at each edge point (x, y, z), $C_\sigma(x, y, z)$ is set to the value of the extremality criterion and to 0 otherwise ;

5. Determination of an image $Z_\sigma(x, y, z)$ set to 1 at each edge point being a zero-crossing of the extremality criterion and to 0 otherwise.

The last stage of this algorithm consists of finding the zero-crossings of a function defined on the discrete trace of a surface (traced by the 3D edge points) which is a difficult task in itself. So far, we have only implemented simple strategies to extract these zero-crossings. But, in order to be solved properly, this delicate problem needs more attention. An interesting solution can be found in [21].

Therefore, the final output of our algorithm is an image Z_σ representing the set of edge points which are zero-crossings of the extremality criterion. Each value of σ defines an image Z_σ representing the crest line for the scale defined by σ.

4 Multi-scale approach to extract crest lines in 3D volumic images

4.1 Why a multi-scale approach ?

The use of the filters presented in section 2 yields to obtain curvatures invariant by a rigid motion which was not exactly the case with the filters presented in [13]. This improves the

quality of the results, but it may not be enough to provide good results in noisy data. As we have seen in the previous section the result of our algorithms is an image Z_σ where the zero-crossing of the extremality criterion are marked. Z_σ defines the crest lines for the scale defined by σ (see section 2). Generally, we see that :

- for simple data, we can obtain good results using a single value for σ but we do not know how to find this value ;
- for more complex data the suitable value for σ varies depending on the area of the 3D image ;
- for noisy data, only the crest lines that can be seen using different scales define reliable features.

Therefore, similar to the edge detection [22] and to the crest line extraction in depth maps [16], it is of great interest to use a multiscale approach. Moreover the recursive implementation of our filters makes it reasonable in terms of computational cost.

4.2 How to merge the results ?

In order to merge the results obtained at different scales $\sigma_i, i = 1, n$ we propose a practical and efficient data structure that we will call the Multi-scale Adjacency Graph : $G_{\sigma_1,\sigma_2,\ldots\sigma_n}$. $G_{\sigma_1,\sigma_2,\ldots\sigma_n}$ is a valuated graph built as follows :

1. each node of $G_{\sigma_1,\sigma_2,\ldots\sigma_n}$ is attached to a point (i,j,k) such that for at least one scale σ_m we have $Z_{\sigma_m}(i,j,k) = 1$;
2. the features attached to each node are :
   (a) the coordinates of the corresponding 3D point $((i,j,k))$;
   (b) the values of the scales for which this point is a crest point (all the σ_p such that $Z_{\sigma_p}(i,j,k) = 1$) ;
   (c) the differential characteristics extracted for all the scales : principal curvatures and principal curvatures directions, value of the extremality criterion.

3. we define an edge joining two nodes of $G_{\sigma_1,\sigma_2,...\sigma_n}$ if and only if the two corresponding points are adjacent for the 26-connectivity ;

Therefore $G_{\sigma_1,\sigma_2,...\sigma_n}$ represents the results of the crest points extraction for the different scales and their spatial relationships. This data structure is particularly efficient when the stability of the crest point locations through different scales is a good selection criterion. Our experiments performed on real and synthetic data show that generally the position of the reliable crest points remain the same for different values of the scale σ (i.e. the shifts of the crest points are less than one pixel).

For instance, the following simple pruning strategy for the graph $G_{\sigma_1,\sigma_2,...\sigma_n}$ can be used :

1. select all nodes corresponding to points which are crest points for at least a given number of scales ;

2. select the connected components having at least a given number of nodes (this threshold corresponds to the minimal number of points of a crest line).

4.3 Algorithm

We come up with the following algorithm :

1. Computation of the zero-crossings of the extremality criterion for a given set of scales : $\sigma_1, \sigma_2, ...\sigma_n$; the result is a set of images $Z_{\sigma_1}, Z_{\sigma_2}...Z_{\sigma_n}$ (see section 3.5) ;

2. building of the multi-scale graph $G_{\sigma_1,\sigma_2,...\sigma_n}$ (see section 4.2) ;

3. pruning of $G_{\sigma_1,\sigma_2,...\sigma_n}$ to select reliable crest points.

5 Experimental results

We present experimental results obtained on synthetic and real data from the implementation of the algorithms described in the previous section

Figures 4 to 7 present results obtained for a synthetic ellipsoid. Here the multiscale approach allows to select only the right crest points. We have extracted the extrema of the

maximum curvature in the maximum curvature direction, and the extrema of the minimum curvature in the minimum curvature direction. It implies that some spurious points can be detected, but it especially enables us to extract the three crest lines of the ellipsoid, which is impossible if we only extract the extrema of the maximum curvature. For the elliptic cylinder, we have extracted the extrema of the maximum curvature in the maximum curvature direction.

We have tested our method on two 3D X-ray scanner images of the same skull taken at two different positions (see figures 9 to 20). In that case, we have chosen to extract only the extrema of the maximum curvature in the maximum curvature direction. The stability of the results we obtain for a single scale illustrates the rotation invariance of our computation of the curvatures and of the extremality criterion (see section 2). We also show that the multiscale scheme allows to remove many spurious crest points. We notice that the result obtained by thresholding the maximum curvature is acceptable for some scales but depends completely on the threshold. On the other hand the results provided by the zero-crossing of the extremality criterion may seem worse for some scales but do not depend on any threshold.

We point out that the size of the convolution mask for a direct implementation of a 3D Gaussian of variance σ^2 is $(8\sigma)^3$ (for $\sigma = 4$ we obtain 8192 !). The use of recursive filters of order 4 reduces this computational cost to about 100 operations per point for any value of σ. Of course, the previous remark applies also for the derivatives of the gaussian filter. Therefore, even for a single space scheme, the recursive filtering appears as a crucial tool.

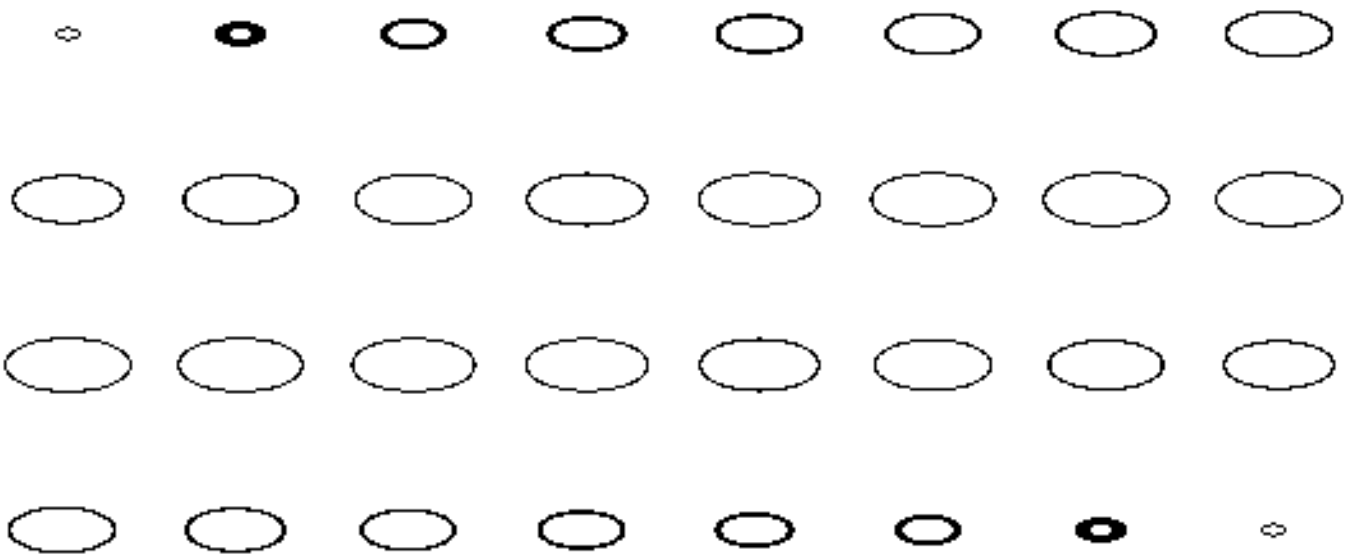

Figure 4: Cross sections of the 3D edge map of an ellipsoid.

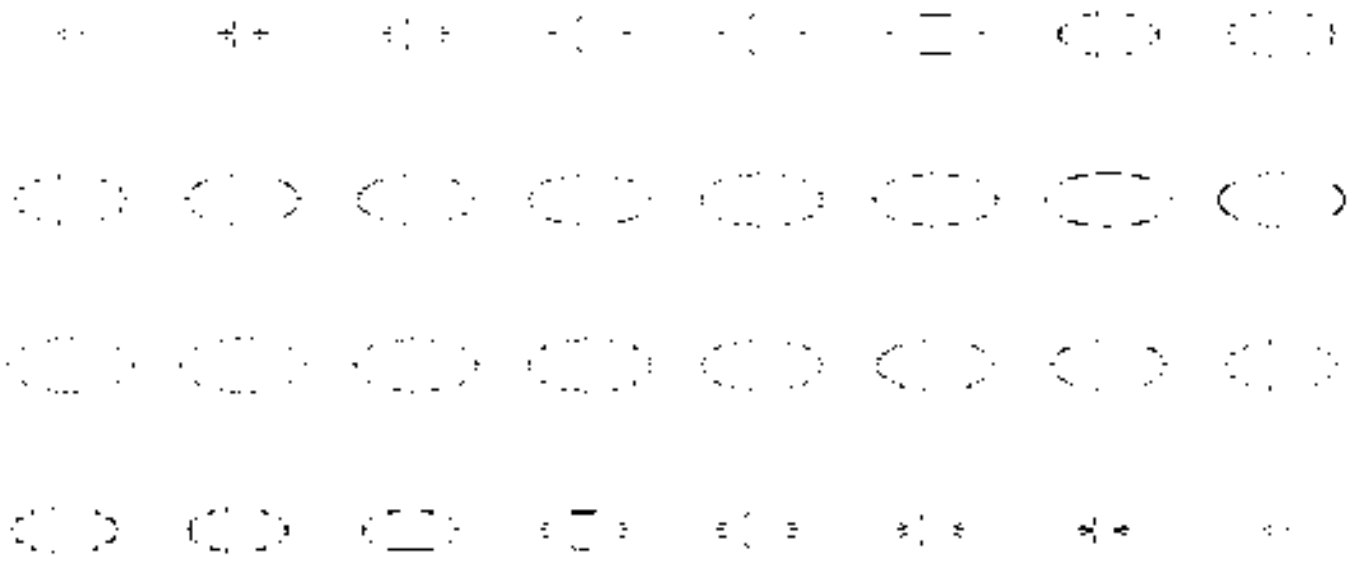

Figure 5: Cross sections of the crest point map of the ellipsoid ($\sigma = 3$).

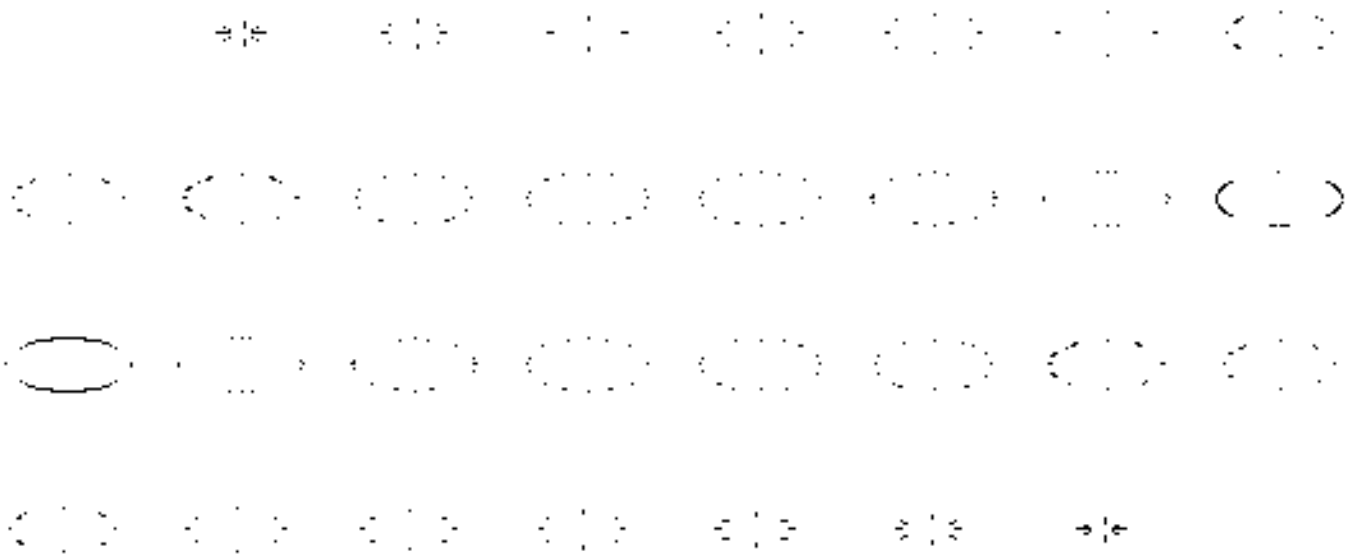

Figure 6: Cross sections of the crest point map of the ellipsoid ($\sigma = 7$).

Figure 7: Cross sections of the crest point map of the ellipsoid when only the stable crest points through the different scales ($\sigma = 1, 2, 3, 4, 5, 6, 7, 8, 9, 10$) are selected. We show the points which appear at at least 6 scales.

Figure 8: Cross sections of the crest point map of the ellipsoid when only the crest points present at each scale are selected. These points correspond to the maxima of the maximum curvature, which appear to be the most stable points.

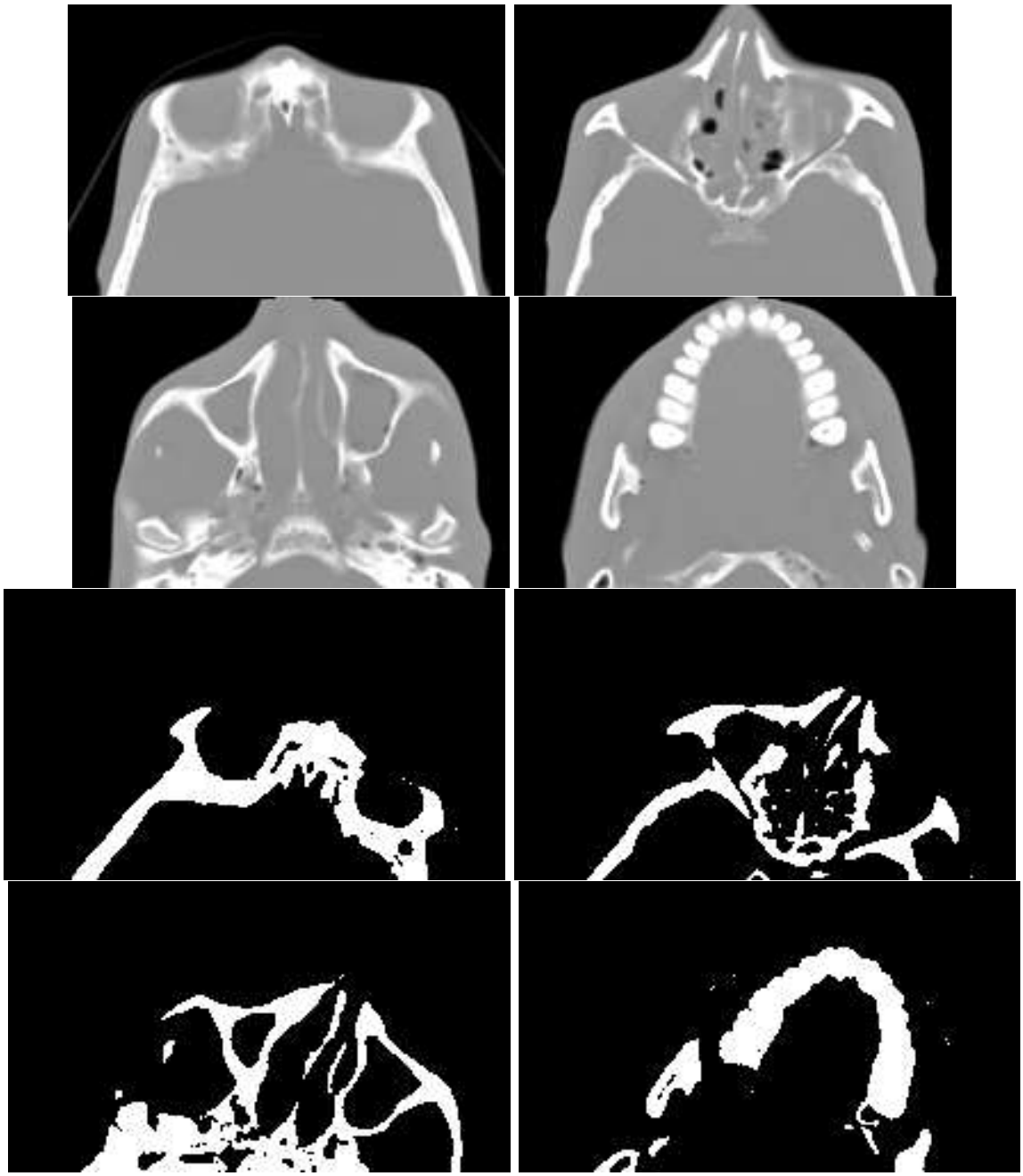

Figure 9: Cross sections of two X-scanner images of the same skull with two different positions A and B (sizes of the image : 192.128.151 for the position A and 220.128.148 for the position B). Up : cross sections corresponding to position A, bottom : cross sections corresponding to position B.

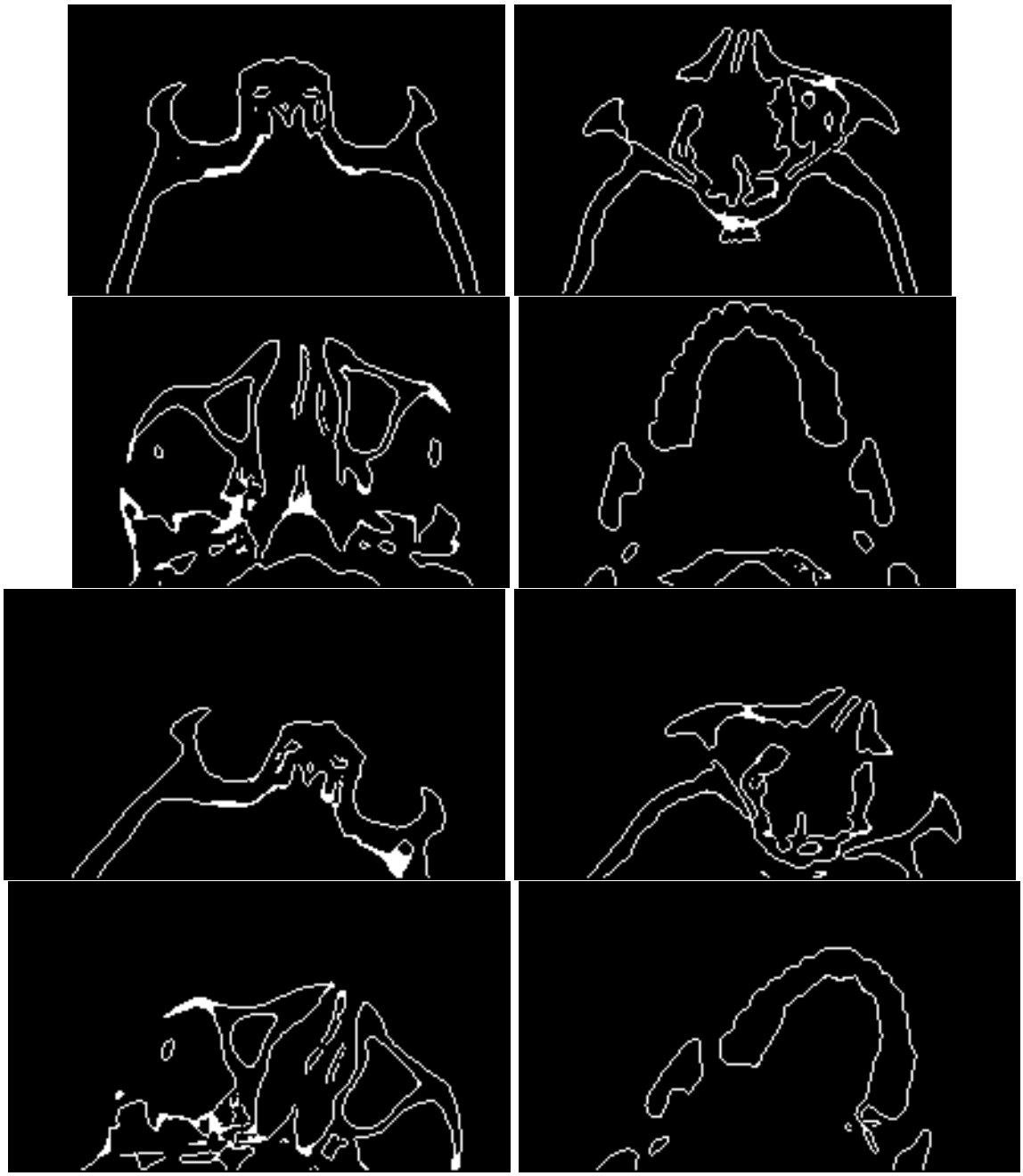

Figure 10: Cross sections of the 3D edge map corresponding to the previous figure). Up : cross sections corresponding to position A, bottom : cross sections corresponding to position B.

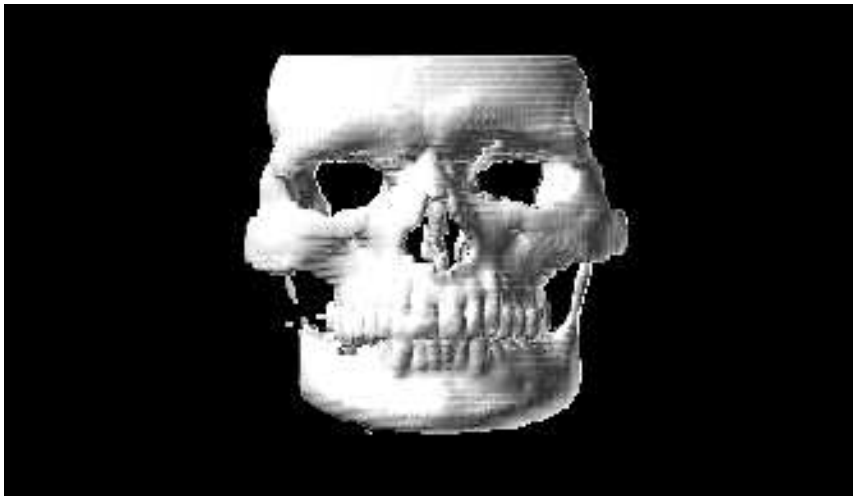

Figure 11: Perpective view of the 3D edge map for the position A

Figure 12: From left to right perpective views of : the maximum curvature, the high maximum curvature points, the zero-crossings of the extremality criterion : cross sections corresponding to position A; σ is set to 1 ; for this figure and the two following ones we only show the upper part of the skull.

Figure 13: From left to right perpective views of : the maximum curvature, the high maximum curvature points, the zero-crossings of the extremality criterion : cross sections corresponding to position A; σ is set to 3.

Figure 14: From left to right perpective views of : the maximum curvature, the high maximum curvature points, the zero-crossings of the extremality criterion : cross sections corresponding to position A; σ is set to 5.

Those results show that the multi-scale approach is much more efficient if we extract the zero-crossings of the extremality criterion instead of the high maximum curvature points. The smoothing removes many spurious points in the results coming from the high maximum curvature point extraction, but makes the crest lines thicker. On the contrary, the lines coming from the detection of the zero-crossings of the extremality criterion are still precise if σ is set to a high value, even if some points are removed.

We now show the images of the zero-crossings of the extremality criterion computed on the entire skull, for the positions A and B and for four values of σ.

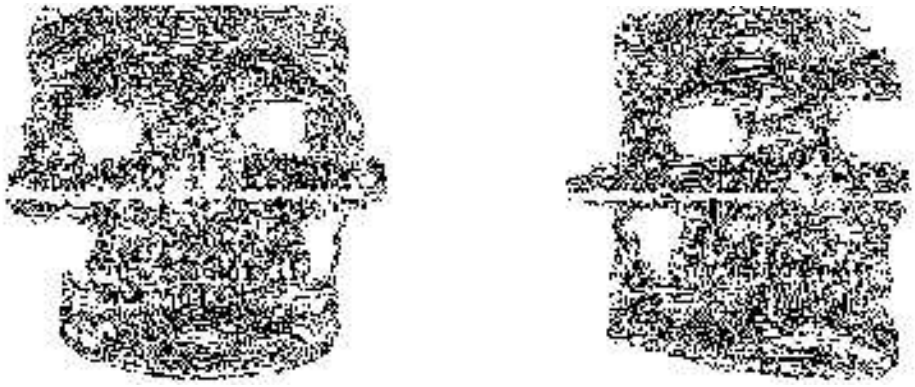

Figure 15: σ is set to 1

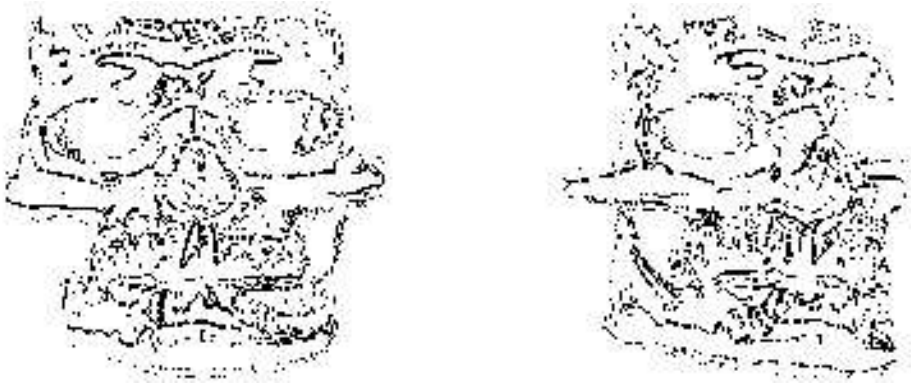

Figure 16: σ is set to 3

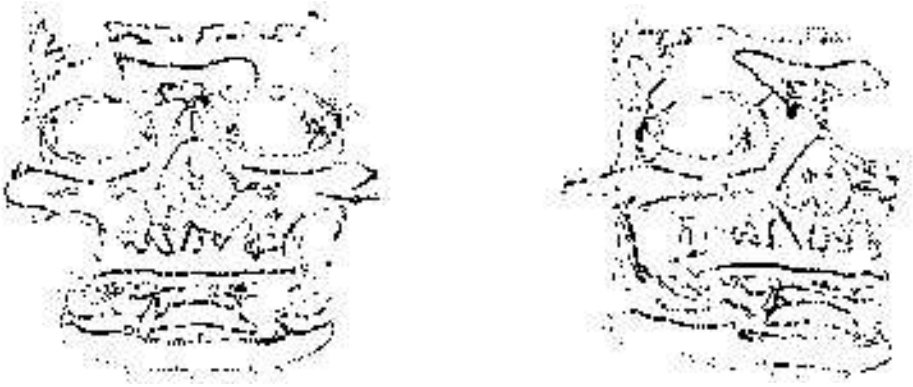

Figure 17: σ is set to 5

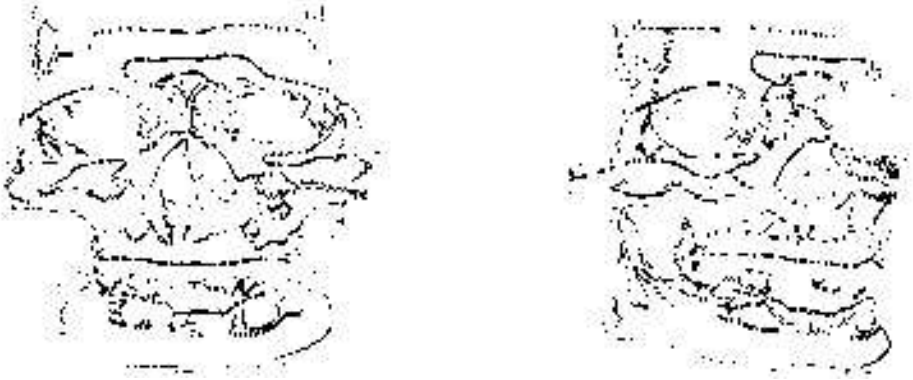

Figure 18: σ is set to 7

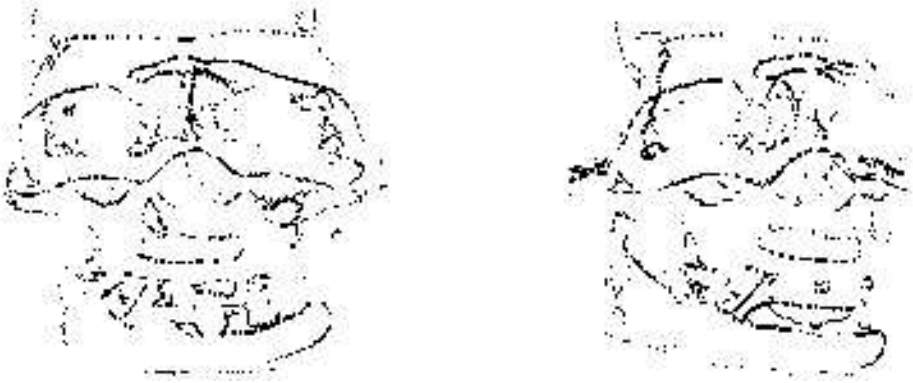

Figure 19: σ is set to 9

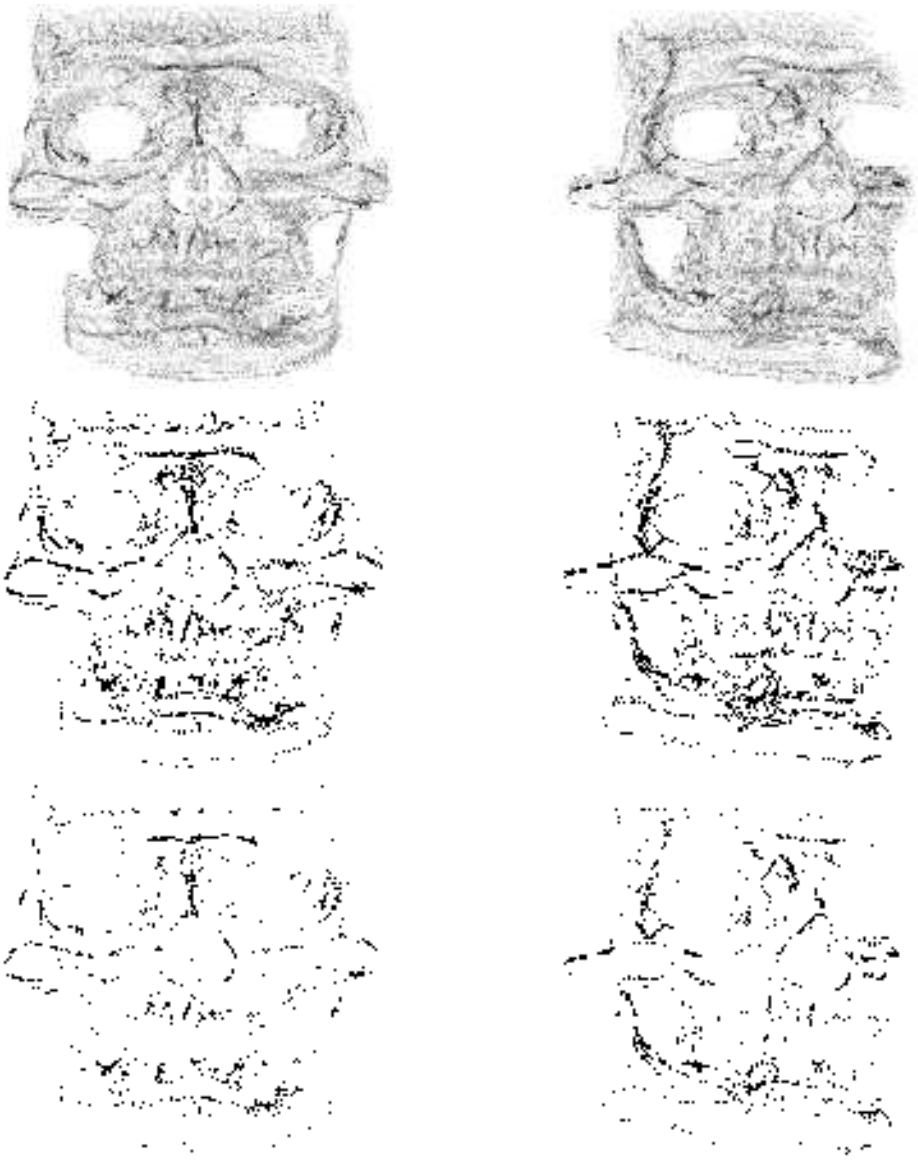

Figure 20: Up : perspective views corresponding to the positions A (left) and B (right) where the grey level is set to the number of scales such that the point is a crest point ; middle : perspective views corresponding to the positions A (left) and B (right) where only the points which are crest points for at least 4 scales are marked; bottom :perspective views corresponding to the positions A (left) and B (right) where only the points which are crest points for at least 5 scales are marked.

6 Conclusion

We have presented a multi-scale approach to extract crest lines of the surfaces represented by 3D volumic images. Compared to the method described in [13] we have developed the following points :

- we show the great theoretical interest in using filters derived from an isotropic smoothing filter to compute partial derivatives of an image ;
- we propose to use recursive filters approximating the Gaussian and its derivatives to obtain differential characteristics invariant by rigid motion ;
- in order to improve the stability of the computation of the differential characteristics (curvatures, derivative of the curvature) we use a multi-scale approach.

Moreover, we present experimental results on synthetic and real data. We stress that the same sketch could be used to extract other differential singularities such as : parabolic lines, umbilic points... Besides, this methodology could also be used in 2-D images like interior scenes, to extract corners for instance (see Appendix B).

Appendix A: the approximation and recursive implementation of the gaussian and its derivatives

The gaussian $g(x) = e^{-\frac{x^2}{2\sigma^2}}$ and each of its derivatives up to the third order has to be approximated in a mean square error sense by a function $h(x)$ such that (see [5]):

$$h(x) = \sum_{i=1}^{m} \alpha_i e^{-\frac{\lambda_i x}{\sigma}}$$

where the coefficients α_i and λ_i are conjugate complex numbers (in the present case, m is set equal to 4). The mean square error is defined by:

$$\epsilon^2 = \frac{\sum_{k=0}^{10\sigma}(h(k) - u(k))^2}{\sum_{k=0}^{10\sigma} u(k)^2}$$

where u denotes the function we want to approximate with h.

This problem can be solved using a numerical minimization routine. However, a starting point to the iterative method implemented by that routine can be found using Prony's approximation method.

Prony's method uses an equidistant sampling of the function $((x_0, y_0 = f(x_0)), ..., (x_n, y_n = f(x_n)))$. We can set $x_r = x_0 + rh$, $c_r = \alpha_r e^{\lambda_r x_0}$ and $v_r = e^{h\lambda_r}$, for $r = 0, ...m$, and write $f(x) = \sum_{i=1}^{m} \alpha_i e^{\lambda_i x}$. Consequently, we have the following system:

$$\begin{cases} c_1 + c_2 + ... + c_m = y_0 \\ c_1 v_1 + c2 v_2 + ... + c_m v_m = y_1 \\ \vdots \\ c_1 v_1^m + c_2 v_2^m + ... + c_m v_m^m = y_m \end{cases} \tag{4}$$

If we construct the polynomial:

$$\prod_{i=1}^{m}(v - v_i) = \sum_{i=1}^{m} s_i v_{m-i} = \phi(v) \tag{5}$$

with $s_0 = 1$, multiply each equation of (4) in turn by s_i, $i = 0...m$, and add each of these new equations, we obtain:

$\sum_{i=1}^{m} \phi(v_i)c_i = \sum_{i=0}^{m} s_i y_{m-i} = 0$, since $\phi(v_r) = 0$. If we choose $n \geq m$, we can also write:

$$\begin{cases} y_{m-1}s_1 + y_{m-2}s_2 + ... + y_0 s_m = -y_m \\ y_m s_1 + y_{m-1}s_2 + ... + y_1 s_m = -y_{m+1} \\ \vdots \\ y_{n-1}s_1 + y_{n-2}s_2 + ... + y_{n-m} s_m = -y_n \end{cases}$$

Those $n-m+1$ equations with m unknowns s_i can then be solved by a least-square method. Afterwards, we get the unknowns v_i with solving (5). Finally, we obtain the coefficients λ_i with the formula $\lambda_i = \frac{\ln v_i}{h}$, the c_i from (4), and the α_i from $\alpha_i = c_i v_i^{-\frac{x_0}{h}}$.

The advantage of this method is that we need not set σ to a particular value to compute the coefficients λ_i and α_i. The coefficients a_0, a_1, b_0, b_1, c_0, c_1, ω_0, ω_1 are easily derived from the α_i and λ_i.

From the formulas of Section II.4.3, the normalization constants are expressed as follows:

We define γ_1, γ_2, γ_3, γ_4 as:

- $\gamma_1 = \sum_{k=0}^{\infty} e^{-\frac{k^2}{2\sigma^2}}$
- $\gamma_2 = \sum_{k=0}^{\infty} k^2 e^{-\frac{k^2}{2\sigma^2}}$
- $\gamma_3 = \sum_{k=0}^{\infty} k^4 e^{-\frac{k^2}{2\sigma^2}}$
- $\gamma_4 = \sum_{k=0}^{\infty} k^6 e^{-\frac{k^2}{2\sigma^2}}$.

Empirically, we can show that a function $F(\sigma) = \sum_{k=1}^{\infty} k^\alpha e^{-\frac{k^2}{(2\sigma^2)}}, \alpha > 0$, can be approximated with a very low error by a polynomial function $F_a(\sigma) = I(\alpha)\sigma^{\alpha+1}$, with $I(\alpha) = \int_0^\infty t^\alpha e^{-\frac{t^2}{2}} dt$.
We know that $I(0) = \sqrt{\frac{\pi}{2}}, I(1) = 1$, and $I(\alpha+2) = (\alpha+1)I(\alpha)$.
If $\alpha = 0$, we can just approximate $F(\sigma)$ with $F_a(\sigma) = 1.25\sigma - 0.5$ for $\sigma \geq 0.5$.
Therefore, we get the following approximations:

- $\gamma_1 \approx 0.5 + 1.25\sigma$

- $\gamma_2 \approx \sigma^3\sqrt{\frac{\pi}{2}}$
- $\gamma_3 \approx 3\sigma^5\sqrt{\frac{\pi}{2}}$
- $\gamma_4 \approx 15\sigma^7\sqrt{\frac{\pi}{2}}$

The normalization coefficients are equal to:

- $\beta_0 = \frac{1}{2\gamma_1 - 1} \approx \frac{0.4}{\sigma}$
- $\beta_1 = -\frac{1}{2\gamma_2} \approx -\frac{0.4}{\sigma^3}$
- $\beta_2 = -\frac{2\gamma_2}{-2\gamma_2^2 + 2\gamma_3\gamma_1 - \gamma_3} \approx -\frac{0.4}{\sigma^3}$
- $\beta_3 = \frac{2\gamma_1 - 1}{2\gamma_2} \approx \frac{1}{\sigma^2}$
- $\beta_4 = \frac{3}{\frac{\gamma_4\gamma_2}{\gamma_3\sigma^2} - \frac{\gamma_3}{\sigma^2}} \approx \frac{1.2}{\sigma^3}$
- $\beta_5 = \frac{3\gamma_2}{\gamma_3(\frac{\gamma_4\gamma_2}{\gamma_3\sigma^2} - \frac{\gamma_3}{\sigma^2})} \approx \frac{0.4}{\sigma^5}$

We have the following results for the coefficients of the approximated filter:

	$g(x) = \beta_0 e^{-\frac{x^2}{2\sigma^2}}$	$df(x) = \beta_1 x e^{-\frac{-x^2}{2\sigma^2}}$
a_0	$0.657/\sigma$	-0.173/σ^2
a_1	$1.979/\sigma$	-2.004/σ^2
c_0	-0.258/σ	$0.173/\sigma^2$
c_1	-0.239/σ	$0.444/\sigma^2$
b_0	1.906	1.561
b_1	1.881	1.594
ω_0	0.651	0.700
ω_1	2.053	2.145

For g and $\sigma = 1$, $\epsilon^2 = 9.7E-09$. For df, $\epsilon^2 = 1.2E-06$.

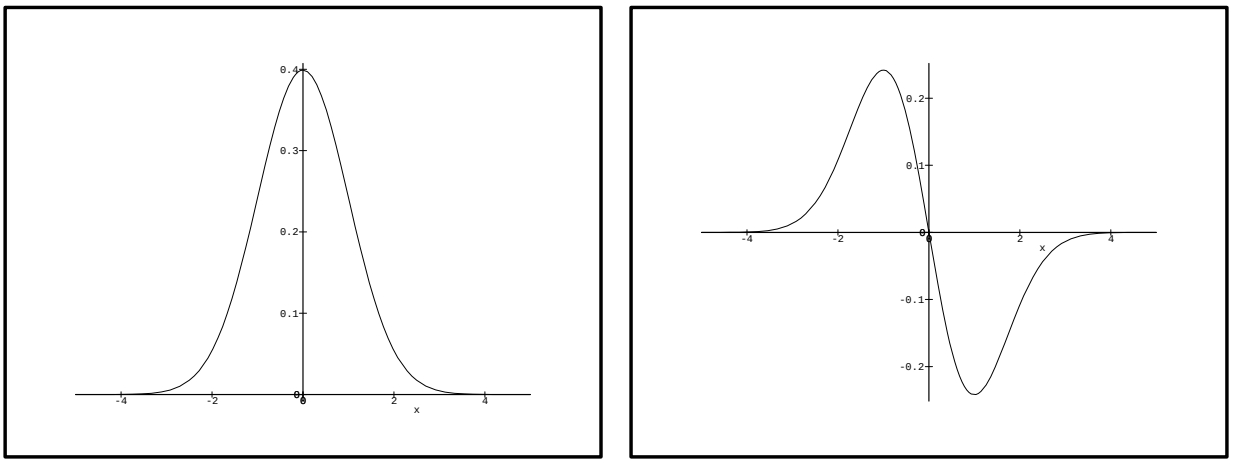

Figure 21: The normalized gaussian filter and its first derivative ($\sigma = 1$)

For $d2f(x) = \beta_2(1 - \beta_3 x^2)e^{-\frac{x^2}{2\sigma^2}}$ and $d3f(x) = \frac{1}{\sigma^2}(\beta_4 x - \beta_5 x^3)e^{-\frac{x^2}{2\sigma^2}}$, we obtain:

	$d2f(x)$	$d3f(x)$
a_0	-0.724/σ^3	1.286/σ^4
a_1	1.689/σ^3	-0.290/σ^4
c_0	0.325/σ^3	-1.286/σ^4
c_1	-0.721/σ^3	0.262/σ^4
b_0	1.295	1.012
b_1	1.427	1.273
ω_0	0.779	0.947
ω_1	2.234	2.338

For $d2f$ and $\sigma = 1$, $\epsilon^2 = 7.3E-06$. For $d3f$ and $\sigma = 1$, $\epsilon^2 = 1.9E-04$.

Once the coefficients $a_0, a_1, c_0, c_1, b_0, b_1, \omega_0, \omega_1$ are computed, we can easily implement the function $h(n)$ in a recursive way. We compute the z-transform $H_1(z) = \sum_{n=0}^{\infty} h(n)z^{-n}$ for the causal part of the filter, and $H_2(z) = \sum_{n=-\infty}^{-1} h(n)z^{-n}$ for its anticausal part. The details of the implementation can be found in [5].

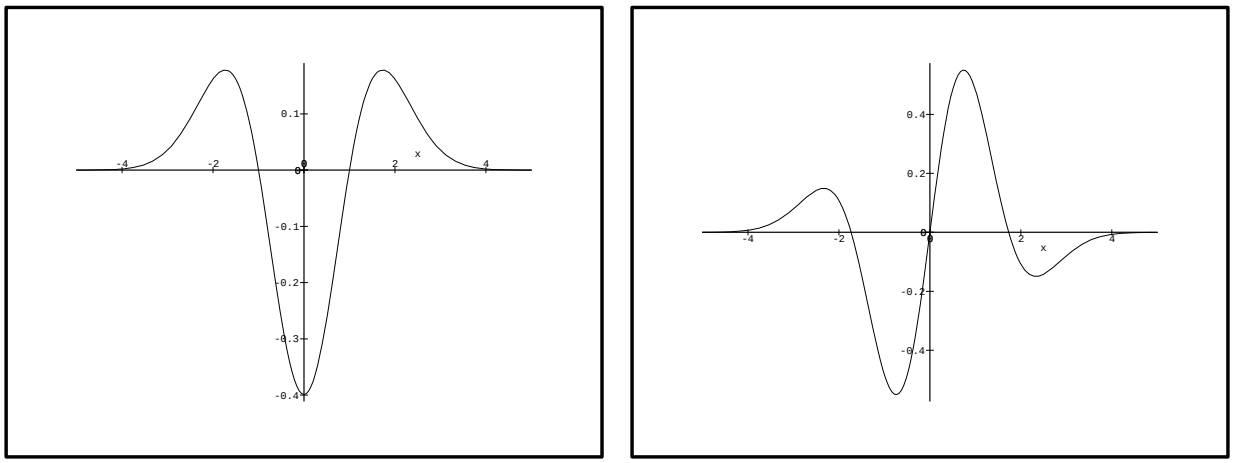

Figure 22: The normalized second and third derivative of the gaussian filter ($\sigma = 1$)

Appendix B: a multi-scale approach for 2-D images

We give here an example of the results obtained by running our algorithm with 5 scales ($\sigma = 1, 2, ..., 5$) and selecting the extrema appearing at at least 2 scales. Those extrema appear in white on the image. We can then extract the corners by selecting the points with the curvature value (it can be done easily if using the graph structure).

References

[1] Nicholas Ayache. Computer vision applied to 3d medical images : Results, trends and future challenges. *INRIA Report research*, September 1993.

[2] J. Koenderink B. Romeny, L. Florack. Invariant third order properties of isophotes: T-junction detection. In *Proc. 7th Scandinavien Conference on Image Analysis, Aalborg*, August 1991.

[3] John Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-8(6):679–698, November 1986.

[4] R. Deriche. Fast algorithms for low level vision. *IEEE Transactions on Pattern Analysis and machine Intelligence*, 1989.

[5] R. Deriche. Recursively Implementing the Gaussian and Its Derivatives. In *Proc. Second International Conference On Image Processing*, pages 263–267, Singapore, September 7-11 1992. A longer version is INRIA Research Report RR-1893.

[6] Rachid Deriche. Using Canny's criteria to derive a recursively implemented optimal edge detector. *International Journal of Computer Vision*, pages 167–187, 1987.

[7] Richard Gordon, Gabor T. Herman, and Steven A. Johnson. Image reconstruction from projections. *Scientific American*, 233(4):56–68, October 1975.

[8] Gabor T. Herman and Hsun Kao Liu. Three-dimensional display of human organs from computed tomograms. *Computer Graphics and Image Processing*, 9:1–21, 1979.

[9] Jan J. Koenderink. *Solid Shape*. MIT Press, Boston, 1990.

[10] O. Monga, N. Ayache, and P. Sander. From voxel to intrinsic surface features. *Image and Vision Computing Volume 10, Number 6*, Aout 1992. a shortened version is in IPMI'91, Lecture Notes in Computer Science 511, Springer Verlag.

[11] Olivier Monga, Nicholas Ayache, and Peter Sander. From voxel to curvature. In *IEEE Conference on Vision and Pattern Recognition*, Hawaii, June 1991.

[12] Olivier Monga and Serge Benayoun. Using differential geometry in r^4 to extract typical surface features. In *IEEE Conference on Vision and Pattern Recognition*, New York, June 1993.

[13] Olivier Monga, Serge Benayoun, and Olivier D. Faugeras. Using third order derivatives to extract ridge lines in 3d images. In *IEEE Conference on Vision and Pattern Recognition*, Urbana Champaign, June 1992.

[14] Olivier Monga, Rachid Deriche, and Gregoire Malandain. Recursive filtering and edge closing: two primary tools for 3d edge detection. *Image and Vision Computing, Vol. 9, Number 4*, August 1991. A shortened version is in proc. of ECCV'90, Lecture notes in Computer Science 427.

[15] Olivier Monga, Rachid Deriche, and Jean-Marie Rocchisani. 3d edge detection using recursive filtering: Application to scanner images. *Computer Vision Graphic and Image Processing, Vol. 53, No 1, pp. 76-87*, January 1991.

[16] Jean Ponce and Michael Brady. Toward a surface primal sketch. In *Proceedings, IJCAI*, 1985.

[17] Peter T. Sander. Generic curvature features from 3-D images. *IEEE Transactions on Systems, Man, and Cybernetics*, 19(6):1623–1635, November 1989.

[18] Peter T. Sander and Steven W. Zucker. Singularities of principal direction fields from 3-D images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. To appear. Available as Technical Report CIM-88-7, McGill Research Center for Intelligent Machines, McGill University, Montréal.

[19] Peter T. Sander and Steven W. Zucker. Inferring surface trace and differential structure from 3-D images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(9), September 1990.

[20] Micheal M. Ter-Pogossian, Marcus E. Raichle, and Burton E. Sobel. Positron-emmision tomography. *Scientific American*, 243(4):170–181, October 1980.

[21] J-P. Thirion and Gourdon A. The 3d marching lines algorithm and its application to crest lines extraction. Technical Report 1672, INRIA, May 1992.

[22] A. Witkin. A multi-scale approach to extract zero-crossings of the laplacian. In *Proceedings of International Joint Conference on Artificial Intelligence*, 1982.

[23] S.W. Zucker and R.M. Hummel. A three-dimensional edge operator. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-3(3):324–331, May 1981.


INRIA