Fast Computation of the Normal Vector Field of the Surface of a 3D Discrete Object

Alexandre Lenoir, Rémy Malgouyres, Marinette Revenu

Fast Computation of the Normal Vector Field of the Surface of a 3-D Discrete Object

Alexandre LENOIR, Rémy MALGOUYRES and Marinette REVENU

GREYC URA 1526, ISMRA
6 Bd du Maréchal Juin 14050 CAEN Cedex, France
E-Mail : Alexandre.Lenoir@greyc.ismra.fr

Abstract : **The shape description of the surface of three-dimensional discrete objects is widely used for displaying these objects, or measuring some useful parameters. Elementary components of discrete surfaces, called surfels, contain some geometric information, but at a scale that is too small with respect to the scale at which we actually want to describe objects. We present here a fast computational technique to compute the normal vector field of a discrete object at a given scale. Its time cost is proportional to the number of surfels at and little dependent on the scale. We prove that our algorithm converges toward the right value in the case of a plane surface. We also give some experimental results on families of curved surfaces.**

Index Terms - Discrete surfaces, surfels, geometric invariant.

1 Introduction

Modern imaging techniques like MRI or confocal microscopy, produce 3-D digital images from real world scenes. A segmentation step followed by a labelling of the resulting binary image yields well identified 3-D discrete objects. The local orientations of their surface, and their areas characterise these objects. For example, they can be used for registration, recognition and medical diagnosis. We use here discrete surfaces composed of surfels. This kind of surface is still actively studied in arbitrary dimension [4], [9], but its detection, and properties are already well known in 3-D ([6]). The method described here uses the regular structure of the discrete surface as the support of functions of vectorial values that describe the geometry of the surface at the surfel scale. Then, we convolve recursively these values by a low pass filter, in order to get a regional average of these local geometrical values. The result is a geometrical value at a less local scale that describes better the real world object. One of the strength of this method is its low time complexity $O\left(\sigma \cdot \sqrt{n} + n\right)$, where σ is the scale parameter of the calculus, and n the number of surfels of the object.

This paper is organised as follows : basic definitions relative to the discrete surfaces used are first given. Then, the recursive calculus of a convolution product involving a summable function and a periodic function is recalled. Next, the algorithm is described. Afterwards, the proof of the convergence in the case of plane surfaces is given. We show that one pass method leads to errors is some cases. Two alternative methods are then presented, both correcting this failure. The time complexity is then estimated. Then come experimental results on families of analytic objects, as well as some direct applications and perspectives.

2 Notations and basic definitions

2.1 Surfaces of 3-D discrete objects

These definitions are mainly drawn from [4], [6] or [9]. The used notion of discrete surface, made of surfels, has been chosen for its regularity, and its analogy with continuous surfaces.

2.1.1 Main vectors of R^3, Δ, operations involving these vectors

We consider the euclidian vectorial space R^3 and a direct orthonormal basis $(O,\omega_1,\omega_2,\omega_3)$. We will use the sets of indices $I_+^3=\{1,2,3\}$, $I^3=\{0\}\cup I_+^3\cup\{-x|x\in I_+^3\}$, and $I_*^3=I^3/\{0\}$. We define $\forall i\in I_+^3,\delta_i=\omega_i,\delta_{-i}=-\delta_i=-\omega_i$, the null vector δ_0, $\Delta=\{\delta_i|i\in I^3\}$ and $\Delta_*=\{\delta_i|i\in I_*^3\}$. Finally, we define the multiplicative operator $\otimes$ from the vector cross product $\wedge$: $\otimes:\left|\begin{array}{l} I^3\times I^3\to I^3 \\ (x,y)\to z=x\otimes y|\delta_z=\delta_x\wedge\delta_y \end{array}\right.$.

2.1.2 Voxel, binary scene

Let $x\in R^3$. We denote x_i the ith coordinate of x. R^3 is divided into unit cubes called *voxels* by a set of planes orthogonal to the axes $\left\{P_{i,j},i\in I_+^3,j\in Z+\frac{1}{2}\text{ et }P_{i,j}=\{x\in R^3|x_i=j\}\right\}$. We identify each voxel with its central point of Z^3. A *binary scene* of Z^3 is a function $B:Z^3\to\{0,1\}$. We call $B(v)$ the value of the voxel v. We note $1(B)=B^{-1}(1)$ and $0(B)=B^{-1}(0)$.

2.1.3 6-neighbourhood, 18-neighbourhood, n-path, n-object, n-background

Two voxels c and d are said to be *6-connected* iff they share a face, that is $c-d\in\Delta_*$. They are *18-connected* iff they are 6-connected or if they share an edge : $\exists(\delta_1,\delta_2)\in\Delta_*^2$, $\delta_1\neq\delta_2$ and $\delta_1\neq-\delta_2$ and $c-d=\delta_1+\delta_2$. For $n\in\{6,18\}$, an *n-path* of length l $[v_0.v_1,\ldots,v_{l-1},v_l]$ is a sequence of l+1 voxels so that $\forall j\in[0\ldots l-1],v_j$ et v_{j+1} are n-connected. Let E be a set of voxels. Let x and y be members of E. If there exists a n-path from x to y in E, we say that x and y are n-connected in E. A set E of voxels is *n-connected* iff $\forall(a,b)\in E^2$, a and b are n-connected in E. *n-connected components of E* are equivalence classes of the restriction to E of the equivalence relation « to be n-connected ». If B is a binary scene, an *n-object* is an n-connected component of $1(B)$. A *n-background* is an n-connected component of $0(B)$.

2.1.4 Surfel, Surfel type, Surface, Boundary, Border, Bel

A *surfel* is an oriented surface element. A surfel s is identified by the pair (v_1,v_2) of 6-connected voxels, of which it is the common face. Therefore, the vector $n=v_2-v_1=\delta_i\in\Delta_*$ can take six distinct values. The *type of the surfel s* is $T(s)=i$. We will call v_1 the *start voxel* of s, and v_2 its *end voxel*. A *surface* is a set of surfels. The *boundary* of two disjoint of voxels E_1 and E_2 is the set $\beta(E_1,E_2)=\left\{s=(a_1,a_2)\middle\| s\in\Sigma\text{ and }a_1\in E_1\text{ and }a_2\in E_2\right\}$. A *bel* of a binary scene B is a surfel $a=(c,d)$ so that $c\in 1(B)$ and $d\in 0(B)$. The boundary of a binary scene is the set of its bels. A $\kappa\lambda-$border is the boundary of two components, the first one

κ – connected of $1(B)$, the other one λ – connected of $0(B)$. In what follows we will only consider $\kappa\lambda$ – *border*, where κ is 6 and λ is 18. Such a border is proved to be connected, to have a well connected interior and exterior and to have the Jordan property.

2.1.5 Edgel, Edgel's type, Bel adjacency, Surface graph

We call *edgel* of the surfel a the pair $e=(a,\delta_i)$, with $\delta_i \in \Delta_*$ and $|i| \neq |\mathrm{T}(a)|$. $\mathrm{T}(e)=i$ is the *type* of e. We also say that *the surfel* $a=(c,d)$ *meets the surfel* $a'=(c',d')$ *at the edgel* $e=(a,\delta)$ iff a and a' share an edge. The proposition 3.5 of [9] states that for any binary scene any bel b of this scene, and any edgel e of b, there is exactly 1 bel or exactly 3 bels that meet b in e (Fig. 1). Two bels $a=(c,d)$ et $a'=(c',d')$ are said to be *adjacent* if they meet at an edgel $e=(a,\delta)$ and if a' is the only bel that meets a in e, or, in the case in which 3 bels meet a in e, if a' is the bel $(c,c+\delta)$ (see Fig. 1). We can say that *a' is the neighbour of a at e* or *a' is the neighbour of a at δ*. This neighbouring relation define the notion of path on a surface and of connected component of a surface, as well as the notion of *surface graph*. Each surfel has exactly four neighbours (one per edgel).

2.1.6 Slices, Slice contour, Slice contour function

A slice of Z^3 is a set of voxels in which one coordinate is fixed, the two others being free. The slice denoted $Tr_{i,j}$ is the set of voxels whose ith coordinate is j. Let $b=(c,d)$ be a bel of type i. It belongs exactly to two *slice contours* denoted by $CTr_{i,b}$ for $i \in \mathrm{I}_3^+ / \{|\mathrm{T}(b)|\}$. We call i the type of the slice contour CTr_i. These slice contours are images of *slice contour functions*, denoted $FCTr_{i,b}$. The succession of slice contour bels is naturally defined by their adjacency relation and their type : $FCTr_{i,b}: \begin{vmatrix} Z \to B \\ z \to FCTr_{i,b}(z) \end{vmatrix}$ with : $FCTr_{i,b}(0)=b$, $\forall z \in Z$, $t=\mathrm{T}(z)$, and $e=\delta_t \wedge \omega_i$. $FCTr_{i,b}$ is then recursively defined in the following way :

$\begin{vmatrix} FCTr_{i,b}(z+1) \text{ is the adjacent bel of } FCTr_{i,b}(z) \text{ at } e. \\ FCTr_{i,b}(z-1) \text{ is the adjacent bel of } FCTr_{i,b}(z) \text{ at } -e. \end{vmatrix}$

Slice contours of finite objects are periodic lists of adjacent bels whose start voxels are in a same plane. A slice can contain several slice contours as shows Fig. 2 for horizontal slices.

2.2 Recursive calculus of a discrete convolution product

Let P be the class of functions that are positive, even, increasing on $]-\infty,0]$, decreasing in $[0,+\infty[$, with unit norm. Let $g_{|Z}$ be the restriction to Z of $g \in \mathrm{P}$. Let f be a periodic function of Z with values in R. We want $\chi = g_{|Z} * f$. For some functions, the convolution product can be strictly recursively computed. If not, it is most of the time possible to approximate the kernel g_σ by another one g'_σ, whose convolution product is recursively computable. One can see [2] where is explained how to approximate a gaussian kernel by a sum of exponential as well as the recursive implementation of a non causal filter. The k order recursive calculus of

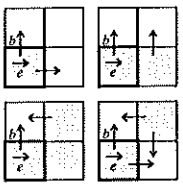

Fig. 1.
Configurations of voxels and bels that meet the bel b at the edgel e. Grey squares are 1-voxels, white one are 0-voxels. Seen in the plane containing b and e.

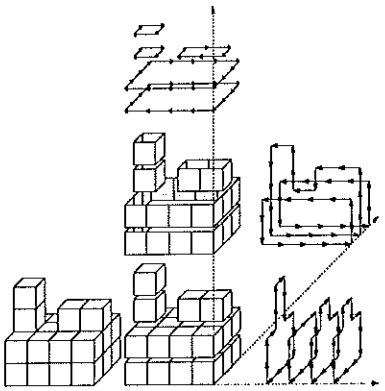

Fig. 2. slices and slice contours of an object along the three families of main planes. Arrows show the orientation of the contours.

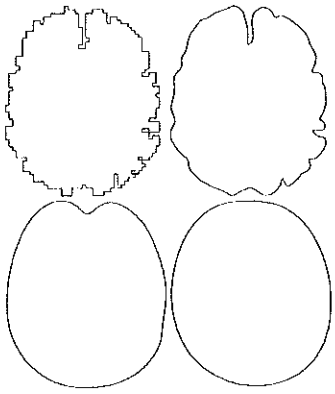

Fig. 3. Geometrical transformation induced by the filtering operation. On each contour, orientation and length has been altered by the filtering : scales used (0, 5, 10, 20). See also [5] or [7]

$\chi = g_{|Z} * f$ is implemented by splitting g into a sum of two functions g^- et g^+, respectively null upon R^+, and R_*^-. For each of these functions, we need only the k previous values of the convolution products as well as the current value of f and its k-1 previous values. At the start of the recursion, we can truncate g outside an appropriate interval and then compute the first k convolutions product in a non recursive way. One can too arbitrarily chose the recursion variables and start the recursion process. The variables will converge toward the right values after some iterations and we will then take into account the results produced. We denote by

$$E(g) = \left\{ g_\sigma \in P \middle| \sigma \in R_+^* : \middle| \begin{matrix} R \to R \\ x \to \frac{1}{\sigma} g\left(\frac{x}{\sigma}\right) \end{matrix} \right\}$$ the family of normalised functions deduced from g by

a scaling factor σ. We denote $g_{|Z}$ the normed restriction of g to Z: $g_{|Z} : \left| \begin{matrix} Z \to R \\ z \to \frac{g(z)}{\sum_{x \in Z} g(x)} \end{matrix} \right.$.

3 Description of our method

We first compute the surface graph of the 6-connected object thanks to a surface tracking algorithm (see [1] or [9]). Then, with a transversal of this graph, the length a well as a start surfel for each slice contour is determined. The next step is to associate to each surfel two elementary geometrical values that solely depends on the type of this surfel and on the slice contour the value correspond to (we recall that any bel belongs exactly to two slice contours). So, we get slice contour functions whose values are no more surfels, but geometrical. We can at least derive two ways to build the slice contour functions, and so to calculate the normal. They will be explained in 5. The recursive calculus of averages of these geometrical values is then done. As seen in 2.2, it is decomposed into two steps : an initialisation step, and a recursive step. We will suppose from here that the surface is planar and orthogonal to the vector (a_1, a_2, a_3), and without loss of generality that $a_1^2 + a_2^2 + a_3^2 = 1$. We will prove in 4.3 that for a discrete plane, the convolution product of any slice contour function converges toward the

average value when the scale of the convolution kernel increases toward infinity. The final step are local computations that compute from the two former values the value of the normal. Depending on the method we choose, we then have to do a second average of the normals in order to avoid drawbacks that occurs in some orientations or a another local computation.

4 Convergence of the method for the surface of a discrete plane

Although the usefulness of the technique is for curved finite discrete object, we will show here that it converge toward the continuous value for any surfel of a discrete plane when the scale increase toward infinity.

4.1 Uniformly distributed functions

Let Ω be the set of functions of $Z \to R$. We denote by Ω_u, $\Omega_u \subset \Omega$ the class of functions whose distribution of values is uniform, called *uniformly distributed* :

$$\forall f \in \Omega_u,\ \exists e>0,\ \exists!\ d=d(f)\in R \mid \forall a \in Z, \forall k \in N,\ d-\frac{e}{k+1} \le \frac{1}{k+1}\cdot\sum_{i=a}^{i=a+k} f(i) \le d+\frac{e}{k+1}.$$

Lemma 1 : Slice contour functions are uniformly distributed for plane surfaces and $d(f)=\dfrac{a_t\cdot h(t)+a_{t'}\cdot h(t')}{a_t+a_{t'}}$ if the surfels present on the contour are of type t and t' and h is a scalar value associated to each surfel type.

4.2 Convergence of the convolution product toward the average for uniformly distributed functions

Lemma 1 : *if f is uniformly distributed and if $g \in P$ then* $\lim_{\sigma\to\infty} \chi = f * g_{\sigma|Z} = d(f)$ *and the convergence speed is in* $\dfrac{1}{\sigma}$.

Proof of lemma 1 : For any $g_{\sigma|Z} | g_\sigma \in E(g)$ we define the list $\left(\alpha\left(g_{\sigma|Z}\right)_i\right)_{i\in N}$ $\alpha\left(g_{\sigma|Z}\right)_i \in R$, $\alpha\left(g_{\sigma|Z}\right)_i = (2i+1)\cdot\left(g_{\sigma|Z}(i)-g_{\sigma|Z}(i+1)\right)$. So we have $\forall i \in Z, g_{\sigma|Z}(i)=\sum_{x=|i|}^{+\infty}\dfrac{\alpha\left(g_{\sigma|Z}\right)_x}{2x+1}$ and $\sum_{i=-\infty}^{+\infty} g_{\sigma|Z}(i)=1$, that is $\sum_{i=-\infty}^{+\infty} g_{\sigma|Z}(i)=\sum_{i=-\infty}^{+\infty}\sum_{x=|i|}^{+\infty}\dfrac{\alpha\left(g_{\sigma|Z}\right)_x}{2x+1}$. If we group the terms in $\left(\alpha\left(g_{\sigma|Z}\right)_i\right)_{i\in N}$ we have $\sum_{i=0}^{+\infty}\sum_{x=-i}^{i}\dfrac{\alpha\left(g_{\sigma|Z}\right)_i}{2i+1}=\sum_{i=0}^{+\infty}\alpha\left(g_{\sigma|Z}\right)_i=1$.

This transformation expresses a cut in horizontal slices of the sum $\sum_{i=-\infty}^{+\infty} g_{\sigma|Z}(i)$. The discrete convolution product $\chi = f * g_{\sigma|Z}$ can then be rewritten in function of the $\left(\alpha\left(g_{\sigma|Z}\right)_i\right)_{i\in N}$:

$$\chi(t)=\left(f*g_{\sigma|z}\right)(t)=\sum_{y=-\infty}^{y=+\infty}\left(f(t-y)\cdot g_{\sigma|z}(y)\right)=\sum_{y=-\infty}^{y=+\infty}\left(f(t-y)\cdot\sum_{x=|y|}^{+\infty}\frac{\alpha\left(g_{\sigma|z}\right)_x}{2x+1}\right)$$

We now group the terms in $\left(\alpha\left(g_{\sigma|z}\right)_i\right)_{i\in N}$. This yields : $\chi(t)=\sum_{y=0}^{y=+\infty}\left(\frac{\alpha\left(g_{\sigma|z}\right)_y}{2y+1}\cdot\sum_{x=t-y}^{t+y}f(x)\right)$.

Moreover, if $f\in\Omega_u$, $\exists e>0$, $\forall s\in Z$, $\left(d(f)-\frac{e}{k}\right)\cdot k\le\sum_{x=s}^{x=s+k-1}f(t-x)\le\left(d(f)+\frac{e}{k}\right)\cdot k$. So, for each term of the sum over y, we have :

$$\alpha\left(g_{\sigma|z}\right)_y\cdot\left(d(f)-\frac{e}{2y+1}\right)\cdot\frac{2y+1}{2y+1}\le\frac{\alpha\left(g_{\sigma|z}\right)_y}{2y+1}\cdot\sum_{x=t-y}^{x=t+y}f(t-x)\le\alpha\left(g_{\sigma|z}\right)_y\cdot\left(d(f)+\frac{e}{2y+1}\right)\cdot\frac{2y+1}{2y+1}$$

$$d(f)\cdot\alpha\left(g_{\sigma|z}\right)_y-\frac{e\cdot\alpha\left(g_{\sigma|z}\right)_y}{2y+1}\le\frac{\alpha\left(g_{\sigma|z}\right)_y}{2y+1}\cdot\sum_{x=t-y}^{x=t+y}f(t-x)\le d(f)\cdot\alpha\left(g_{\sigma|z}\right)_y+\frac{e\cdot\alpha\left(g_{\sigma|z}\right)_y}{2y+1}$$

and by summing over y and applying $\sum_{y=0}^{+\infty}\alpha\left(g_{\sigma|z}\right)_y=1$, we obtain :

$$d(f)-\sum_{y=0}^{+\infty}\frac{e\cdot\alpha\left(g_{\sigma|z}\right)_y}{2y+1}\le\chi(t)=\sum_{y=0}^{+\infty}\left(\frac{\alpha\left(g_{\sigma|z}\right)_y}{2y+1}\cdot\sum_{x=t-y}^{x=t+y}f(t-x)\right)\le d(f)+\sum_{y=0}^{+\infty}\frac{e\cdot\alpha\left(g_{\sigma|z}\right)_y}{2y+1}$$

$$d(f)-e\cdot\sum_{y=0}^{+\infty}\frac{\alpha\left(g_{\sigma|z}\right)_y}{2y+1}\le\chi(t)\le d(f)+e\cdot\sum_{y=0}^{+\infty}\frac{\alpha\left(g_{\sigma|z}\right)_y}{2y+1}$$

We recognise $g_{\sigma|Z}(0)=\sum_{y=0}^{+\infty}\frac{\alpha\left(g_{\sigma|z}\right)_y}{2y+1}$, and the previous expression becomes : $d(f)-e\cdot g_{\sigma|Z}(0)\le\chi(t)\le d(f)+e\cdot g_{\sigma|Z}(0)$.

We now need the following lemma that can be proved by elementary ways :

Lemma 1.1: Let $f\in P$ and $f_\sigma\in E(f)$. We have $\frac{f(0)}{\sigma+f(0)}\le f_{\sigma|Z}(0)\le\frac{f(0)}{\sigma-f(0)}$ or more generally $\forall z\in Z,\frac{\sigma}{\sigma+f(0)}\cdot f_\sigma(z)\le f_{\sigma|Z}(z)\le\frac{\sigma}{\sigma-f(0)}\cdot f_\sigma(z)$.

$$d(f)-e\cdot g_{\sigma|Z}(0)\le\chi(t)\le d(f)+e\cdot g_{\sigma|Z}(0)$$

Finally, this lemma: $d(f)-e\cdot\frac{g(0)}{\sigma-g(0)}\le\chi(t)\le d(f)+e\cdot\frac{g(0)}{\sigma-g(0)}$

The convolution product converges in $\frac{1}{\sigma}$ *toward* $d(f)$.

End of proof of lemma 1.

4.3 Conclusion : Convergence for tangents

The convolution product of vector that depends only on the type of surfels converges in $\frac{1}{\sigma}$ toward their average value. We have seen this for each component of the vector. For example, if these vectors are unit tangents, and then we obtain the tangent to the slice contour.

5 Methods proposed

Two direct methods are first proposed, that appear to give some erroneous results for some surface orientations. We then propose some improvements the correct this drawback.

5.1 One step methods

Two similar methods that uses only one averaging and one local calculus are presented. In the first one, drawn from classical differential geometry, the averaged value depends on the slice contour type, not the local operation. Contrarily, in the second one, the averaged value is the normal to the surfel, for both slice contours, but the final calculus depends on the surfel type.

5.1.1 First method

When two curves on a regular surface intersect at a point P, and are not parallel at this point, then the surface normal is colinear to the vector cross product of their tangents. Hence, an idea is to compute the tangent of each slice contour and then to perform a cross product. More precisely, for each surfel s whose type is t, and each slice contour it belong to, we associate the value $f_{s,j}:\begin{vmatrix} Z \to R^3 \\ z \to \delta_{\otimes(T \circ FCT_{t,j}(z),j)} \end{vmatrix}$ that expresses a elementary move through the surfel when one walks along the slice contour. Therefore, $f_{s,j} * g_\sigma$ converges toward the vector whose jth coordinate is null, and whose coordinate n° $|\otimes(t,j)|$ is $d(f)=\frac{|\otimes(t,j)|}{\otimes(t,j)} \cdot \frac{a_t}{a_t + a_{t'}}$ and whose $|\otimes(t',j)|$th coordinate is $d(f)=\frac{|\otimes(t',j)|}{\otimes(t',j)} \cdot \frac{a_{t'}}{a_t + a_{t'}}$. The same calculation applies for the other slice contour whose type is t'. The vector cross product of the two previous vectors yields, for example with $(t,t',j)=(1,2,3)$:

$$\begin{pmatrix} -\frac{a_3}{a_1+a_3} \\ 0 \\ \frac{a_1}{a_1+a_3} \end{pmatrix} \wedge \begin{pmatrix} \frac{a_2}{a_1+a_2} \\ -\frac{a_1}{a_1+a_2} \\ 0 \end{pmatrix} = \begin{pmatrix} \frac{a_1}{a_1+a_2} \cdot \frac{a_1}{a_1+a_3} \\ \frac{a_1}{a_1+a_3} \cdot \frac{a_2}{a_1+a_2} \\ \frac{a_3}{a_1+a_3} \cdot \frac{a_1}{a_1+a_2} \end{pmatrix} = \frac{a_1}{(a_1+a_2)\cdot(a_1+a_3)} \cdot \begin{pmatrix} a_1 \\ a_2 \\ a_3 \end{pmatrix}.$$

The general expression is $\frac{a_t}{(a_t+a_{t'})\cdot(a_t+a_{t''})} \cdot \begin{pmatrix} a_1 \\ a_2 \\ a_3 \end{pmatrix} = n_t \cdot \begin{pmatrix} a_1 \\ a_2 \\ a_3 \end{pmatrix}$ for a surfel of type is t.

5.1.2 Second method

For each slice contour we calculate a vector contained in the slice plane and that is orthogonal to the contour that pass through the surfel. For both contours the value convolved is the unit normal vector of the surfel. On a surfel of type t, the limit of the convolution product is v or v' depending on the slice contour. For the one whose type is t' the coordinates of v are $v_t = \frac{a_t}{a_t + a_{t''}}$, $v_{t'} = 0$ and $v_{t''} = \frac{a_{t''}}{a_t + a_{t''}}$. For the surfel whose type is t'', the coordinates are $v'_t = \frac{a_t}{a_t + a_{t'}}$, $v'_{t'} = \frac{a_{t'}}{a_t + a_{t'}}$ and $v'_{t''} = 0$. We then compute v'' this way: $v''_t = \frac{v_t \cdot v'_t + v'_t \cdot v_t}{2} = v'_t \cdot v_t$, $v''_{t'} = v_t \cdot v'_{t'} + v'_t \cdot v_{t'} = v_t \cdot v'_{t'}$, $v''_{t''} = v_t \cdot v'_{t''} + v'_t \cdot v_{t''} = v'_t \cdot v_{t''}$. So we have: $v''_t = \frac{a_t \cdot a_t}{(a_t + a_{t'}) \cdot (a_t + a_{t''})}$, $v''_{t'} = \frac{a_{t'} \cdot a_t}{(a_t + a_{t'}) \cdot (a_t + a_{t''})}$ and $v''_{t''} = \frac{a_{t''} \cdot a_t}{(a_t + a_{t'}) \cdot (a_t + a_{t''})}$. The final value obtained is the same as in the first method.

5.1.3 Insufficiency of the one step method in some particular orientations

First, we notice that the frequency of a surfel of type t is proportional to the tth component of the discrete plane normal. We examine here angular errors of the normal calculated with the first one step methods. Tangent vectors will be considered as the sum of two vectors. The first one is the ideal tangent vector of which the 1-norm (sum of the absolute values of the coordinate) is 1, (because the 1 norm of the initial geometrical values are one and the convolution kernel is of norm 1). The second one is an error term e_i. So we have $t_i = T_i + e_i$. Then: $t_i \wedge t_j = (T_i + e_i) \wedge (T_j + e_j) = T_i \wedge T_j + T_i \wedge e_j + T_j \wedge e_i + e_i \wedge e_j = N_{ij} + E$ with $N_{ij} = T_i \wedge T_j$ and $E = T_i \wedge e_j + T_j \wedge e_i + e_i \wedge e_j$. Angular accuracy is an increasing function of $\frac{\|N_{ij}\|}{\|E\|}$. The norm of the normal on surfel of type k is $n_k = \frac{a_k}{(a_k + a_j) \cdot (a_k + a_i)}$. The euclidian norm of each of the tangents is between $\frac{1}{\sqrt{2}}$ and 1, but $\|N_{ij}\|$ can be small if tangents are nearly parallel. This case appears when the surfel of the calculus is of a sparse type. Therefore the one step method leads to a big error if the normal of the plane contains a near zero coordinate, even if the error on tangents is small. Moreover, we have $\frac{n_k}{n_j} = \frac{1 + \frac{a_i}{a_j}}{1 + \frac{a_i}{a_k}}$. Thus, the more frequent in the plane the surfel is, the greater is the norm of the calculated normal, and so the greater is the precision. In discrete planes and straight lines, spare surfels are always isolated. One can show that the norm of the vector cross product calculated on a surfel that is not of the rarest kind is not less than $\frac{1}{\sqrt{6}} = 0{,}408$, and that the one calculated at surfel of the most frequent type is not less than $\frac{\sqrt{3}}{4} = 0{,}433$. We will now infer from these remarks some calculus methods that

guarantee a small angular error whatever the surfel type is.

5.2 Improved one step methods

A simple correction is to check the norm of the normal and, if it is lower than some fixed value, $\frac{1}{\sqrt{6}}$ for example, to take for final value a linear combination of the normal obtained on adjacent surfels, that have certainly a norm greater than $\frac{1}{\sqrt{6}}$. One can also check if the norm ratio between the surfel and some surfels in its neighbourhood is lower than a fixed value.

5.3 Two steps method

Another possibility is to perform another recursive calculus to get an average of the normal obtained at the first step. This is correct because in each of the slice contour, the average norm of the calculated normal $\frac{a_i}{a_i+a_j}\cdot n_i+\frac{a_j}{a_i+a_j}\cdot n_j$ admits the lower bound $\frac{1}{\sqrt{6}}$. The average of this value for the two slice contours $\frac{1}{2}\left(\frac{a_i}{a_i+a_j}\cdot n_i+\frac{a_j}{a_i+a_j}\cdot n_j+\frac{a_i}{a_i+a_k}\cdot n_i+\frac{a_k}{a_i+a_k}\cdot n_k\right)$ has 0,429 for lower bound, for rarest surfels. This value increase for more frequent surfels. This two steps method produces the better experimental results on curved object and will be evaluated in section 7. An essential difference with the previous one step methods is that it take into account not only surfels of the two slice contours, but also those of a 2D neighbourhood around the central surfel. Thus the method is not very sensitive to noise.

The previous corrections guarantee that as the errors made in the calculus on the tangents decrease, then the angular error on the final normal decreases too, for any type of surfel.

6 Evaluation of the algorithmic complexity

If the object is in a cube with n voxels edges, the number of slice contours is roughly $3\cdot n$, and its number of surfels is around n^2. We want to compute the normal field at the scale σ. The cost of the initialization of the recursive calculation is proportional to the number of slice contours and to the scale : $O(\sigma\cdot n)$. The recursive computation needs a constant computational amount for each surfel. There are then only local computations. If the two steps method is used, we just have to double this cost. To conclude, the cost of our method is $O\left(\sigma\cdot n+n^2\right)$.

7 Experimental results and applications

Applications of geometric fields of the surface of an object are numerous and important since such values are the basis of most of the recognition or interpretation processes. Moreover, it is useful to know the limits of the calculus algorithms used, especially when the values computed are used to measure real world quantities.

7.1 Angular errors

We have done several calculus upon families of spheres and toruses, using the two steps method and varying the scale of the filter used for each step. We have first noticed that the nature of the low pass filter used had little consequence on the quality of results. So, it is possible to use a first order recursive filter that implements the convolution product by the

function $\frac{1}{\sigma} \cdot e^{-\left|\frac{x}{\sigma}\right|}$. Some scales produce best results for mean and maximal angular errors. These scales depend on the surface of the object. For planes surfaces, the larger the scales are, the better the results. For curved surfaces, these optimal scales depend on the curvature of these surfaces. As we want a local value, if we increase the scale, we take into account a wider neighbourhood of surfels and if the distribution of surfels is not uniform, we increase the error. On the other hand, if we decrease the scale, the value on the central surfel becomes predominant, and we get a value that corresponds to the structure that describes the object, but not to the object itself. We notice that for the sphere the average error is smaller than the variation of the normal orientation along a single surfel. The accuracy of the method is at the scale of the surfel for a sphere. But the case of sphere is advantageous since its slice contours are convex, and that we then take advantage of the local symmetry. This is not the case for the torus for which, compared to a sphere with equal curvature we get a greater error. But this does not break the rule : the smaller the curvature, the smaller the angular error, and the greater the scale leading to the minimal error. In other words, we can say that the accuracy increases when the discretization step decreases. The error considered is the angle in degree between the normal calculated by our algorithm and the theoretical normal at the centre point of the surfel.

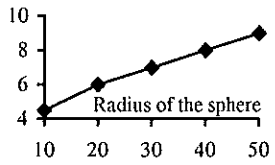

Fig. 4. Scales leading to the minimal maximum angular error (sum of the scale of each step)

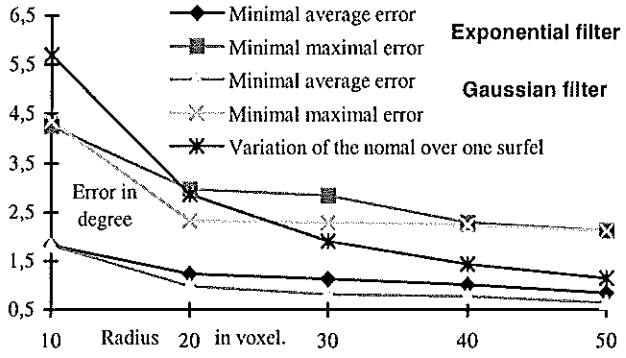

Fig. 5

7.1.1 Results for spheres

For spheres, each orientation is equally represented, and the problem of discretization orientation does not exist. Scales that lead to a minimal error are roughly identical for the two steps. We recall that the first step consist in calculating the tangents of each slice contour passing through a surfel and that the second step produce an average of the normals obtained by cross product. We say that the scale of the calculus is the sum of the scales of each step.

7.1.2 Results for toruses

These toruses are revolution surfaces around the Z axis generated by circle of r_1, called small radius. This circle is in the Oz Ox plane. Its centre is at a distance r_2, called big radius, of the Z axis. The curvature of a r_1 r_2 torus correspond to the one of a r_1 sphere. The results are not as good as those of the spheres because slice contours have inflexion points. Nevertheless, they follow the rule given in 7.1. The value of the minimum error in Fig. 6 is repeated in Fig. 7.

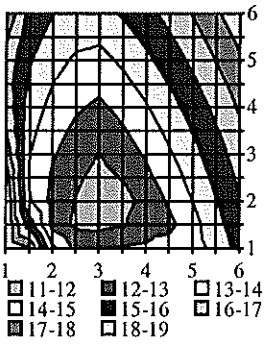

Fig. 6. maximal angular errors for a 10 20 torus for the scale of each step

Little and big radius of the torus.	10 20	20 40
Scales leading to the smaller angular error.	2 3	4 4
maximal and average angular error.	11,41 4,34	7,76 2,84

Fig. 7. Error for toruses.

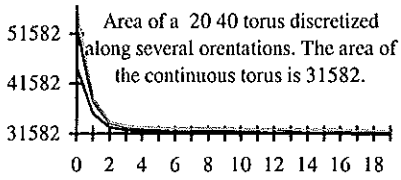

Fig. 8

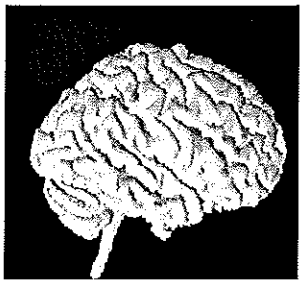

Fig. 9. Shading of a cortical surface using the normal orientations.

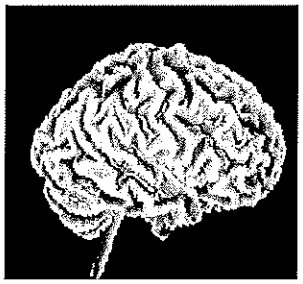

Fig. 10. Normal are computed for two different scales, a global one, that ignore the ridges and valleys of the surface, and a smaller one that take them into account. The surface is brighter where the two orientations correspond.

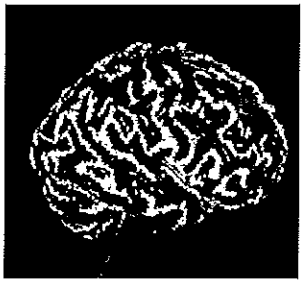

Fig. 11. Thresholding of the previous picture. Cortical sulci and giri appear clearly.

7.2 Area calculus

Another application is the computation of the area of a discrete surface. If we do this by counting the number of surfels, the result for a plane surface may be from 1 to $\sqrt{3}$ times the right value, depending on the orientation of the plane. For a discrete sphere of radius r (the unit is the discretization step), the area will roughly be of $6 \cdot \pi \cdot r^2$, whereas the right value is around $4 \cdot \pi \cdot r^2$. A much better value is obtained by noticing that the contribution of a surfel to the total area is the area of its projection on its normal plane. For spheres, the relative error with this technique is less than 0.1%, and for toruses, around 1%. Since the area of each surfel is constant, the calculus of an area is the sum of the scalar product of the normal of the surface with the normal of each surfel. Since only one component of a normal to a surfel is non zero, we just have to sum the corresponding components of the calculated surface normal.

7.3 Applications

A straightforward application of the normal field of the surface, illustrated Fig. 9 is the display of voxel objects, with object space shading. The precise structure of the object is preserved, and

no smoothing artefact due to image space shading is introduced. It is also possible to compare orientations of two normal fields computed at two different scales to exhibit some details specific to a scale. This is shown on a brain in Fig. 10 and Fig. 11.

8 Conclusion and perspectives

We have described an accurate and efficient technique for compute the normal field of a discrete surface made of surfels. It depends on a scale parameter. Both time and space complexity is linear with respect to the number of surfels. The calculus of the normal field of the surface of an human brain composed of 170000 surfels takes ten seconds on a SUN SPARC 10. We have proved the convergence of the method at each surfel for a plane surface. Our method differs from the ones proposed in [8] where partial derivatives of the grey level image are used. We here just use the discrete surface of the object, a two dimensional structure that has most of the time a far less cardinality than the 3-D volume enclosing the object. Our method is fast enough to be useful in an interactive tool of manipulation of discrete surfaces, or in a multi-scale context. The interest of such a tool should be greatly improved if higher order differential invariants like curvatures were available. We can consider the surface as a classical 2-D grey level image where the grey level is a geometrical value. It is then possible to adapt some operators working on classical 2-D grey level images to do some segmentation work. Our goal is indeed to segment complex surfaces like human cortex surfaces by mean of local geometrical properties. We have seen that the comparison of the local orientation of the surface computed at two different scales could be a good starting point to achieve this project.

9 Bibliography

[1] E. Artzy, G. Frieder, G. T. Herman. The Theory, Design, *Implementation and Evaluation of a Three-Dimensional Surface Detection Algorithm.* CVGIP 15,1-24 (1981).

[2] R. Deriche. *Recursively Implementing the Gaussian and its Derivatives.* Rapport de Recherche de l'INRIA No 1893, Avril 1993.

[3] G. T. Herman H. T. Liu. *Three-Dimensional Display of Human Organs from Computed Tomograms.* Computer Graphics and Image Processing 9. 1-21, 1979.

[4] G.T. Herman. *Discrete Multidimensional Jordan Surfaces.* CVGIP : Graphical Models and Image Processing. Vol 54 No 6, November, pp 507-515, 1992.

[5] F. Mokhtarian A. Mackworth *Scale-Based Description and Recognition of Planar Curves and Two-Dimensional Shapes.* IEEE Transactions on Pattern Analysis and Machine Intelligence PAMI-8 No 1 January 1986, pp. 665-668.

[6] A. Rosenfeld, T. Y. Yung Kong, A. Y. Wu *Digital Surfaces.* CVGIP 53, No 4, July, pp 305-312, 1991

[7] G. Sapiro A. Tannenbaum *Area and Length Preserving Geometric Invariant Scale-Spaces.* IEEE Transactions on Pattern Analysis and Machine Intelligence Vol.-17 No 1 January 1995, pp.67-72.

[8] J.P. Thirion. *New Feature Points based on Geometric Invariants for 3D Image Registration.* Rapport de Recherche de l'INRIA N° 1901, Mai 1993.

[9] J.K. Udupa. *Multidimensional Digital Boundaries* in CVGIP: Graphical Models and Image Processing. Vol 56 N° 4. July 1994, pp. 311-323.

[10] T. J. Fan G. Medioni, R. Nevata. *Recognising 3D Objects Using Surface Descriptions.* IEEE Transactions on Pattern Analysis and Machine Intelligence Vol.-11 No 11 November 1989, pp. 1140-1157.