Taylor & Francis
Taylor & Francis Group

Surface approximation using growing self-organizing nets and gradient information

doi:10.1080/11762320701797745

Jorge Rivera-Rovelo and Eduardo Bayro-Corrochano

Department of Electrical Engineering and Computer Sciences, CINVESTAV del IPN, Unidad Guadalajara, Av. Científica 1145, El Bajío, Zapopan, Jalisco, 45010, México

Abstract: In this paper we show how to improve the performance of two self-organizing neural networks used to approximate the shape of a 2D or 3D object by incorporating gradient information in the adaptation stage. The methods are based on the growing versions of the Kohonen's map and the neural gas network. Also, we show that in the adaptation stage the network utilizes efficient transformations, expressed as versors in the conformal geometric algebra framework, which build the shape of the object independent of its position in space (coordinate free). Our algorithms were tested with several images, including medical images (CT and MR images). We include also some examples for the case of 3D surface estimation.

Key words: Segmentation, self-organizing neural networks, gradient vector flow, geometric algebra, 2D and 3D reconstruction.

Introduction

Medical image processing is an area receiving a lot of attention; particularly, there are several proposals for medical image segmentation: Prastawa *et al.* (2003) present a method for brain tumor segmentation based on abnormalities detection when comparing the patient's image against a digital brain atlas; Andrade (2004) preprocess the image to eliminate the noise, then highlights the contours and selects manually some initial points to finally use a region growing technique for segmentation; Xu (1999) presents a snake called *GGVF-Snake* which guides the curve evolution to the object contour, and shows how to use it to reconstruct the brain cortex; see Moon *et al.* (2002a, 2002b) for further works.

The use of neural networks in medical image processing is an area receiving a lot of attention with a variety of applications such as segmentation or classification of tissues. The self-organizing neural networks such as Kohonen's map or self-organizing map (SOM), neural gas (NG) and growing neural gas (GNG, Fritzke, 1995) have been used broadly when we need to preserve the topology of the data (Mehrotra *et al.*, 1997, Angelopoulou *et al.*, 2005).

In this work we present an approach which uses the *generalized gradient vector flow (GGVF)* (Xu, 1999) to guide the automatic selection of the input patterns, as well as the adaptation process of two self-organized neural networks: a growing version of the SOM and the GNG, to obtain by their training a set of transformations expressed in the conformal geometric algebra framework. These transformations help us define the shape of the object we are interested in. We decided to use such framework because it has the advantage that (rigid body) transformations of geometric entities (e.g., points, lines, planes, circles, spheres) are expressed in compact form as operators called *versors*, which are applied in a multiplicative way to any entity of the conformal geometric algebra. Thus, training the network we do not obtain specific positions for a particular entity (e.g., the positions of points when the weights of the network are interpreted in such a way), but we obtain the transformation that can be applied to entities resulting in the definition of the object contour or its shape.

Note that the authors are proposing a very advanced algorithm using early vision preprocessing and self-organizing neural computing in terms of algebra techniques. We believe that the early vision preprocessing together with self-organizing neurocomputing resembles in certain manner the geometric visual processing in biological creatures. The experimental results show that approach is very promising.

Corresponding Author
Jorge Rivera-Rovelo
Department of Electrical Engineering and Computer Sciences
CINVESTAV del IPN, Unidad Guadalajara
Av. Científica 1145, El Bajío, Zapopan, Jalisco, 45010, México
Email:rivera@gdl.cinvestav.mx

Background

In order to make the article self-contained, let us briefly explain the self-organizing maps and the neural gas algorithms, as well as the geometric algebra.

Self-organizing maps

The self-organizing map is an artificial neural network which is trained using unsupervised learning to produce low-dimensional representation of the training samples while preserving the topological properties of the input space; sometimes it is called *Kohonen's map,* thanks to its inventor, Dr. Teuvo Kohonen. A self-organizing map consists of a single-layer feedforward network where the outputs are arranged in low dimensional grid. Each input is connected to all output neurons. Attached to every neuron is a weight vector of the same dimension as the input vectors.

The adaptation (training) algorithm utilizes competitive learning. When an input (or training example) is fed to the network, its Euclidean distance to all weight vectors is computed. The neuron with weight vector, most similar to the input results in the smallest Euclidean distance, is called the *winner neuron* or the best-matching unit (BMU). Then, the weights of winner and neighboring neurons (neurons close to it in the SOM lattice) are adjusted toward the input vector. This change decreases with time and distance from the winner neuron. The updated formula for a neuron with weight vector at time t is

$$w_v(t+1) = w_v(t) + \phi(v,t)\alpha(t)(\zeta - w_v(t)), \tag{1}$$

where $\alpha(t)$ is a monotonically decreasing adaptation (learning) coefficient, and ζ is the input vector; $\phi(v,t)$ depends on the lattice distance between the winner neuron and its neighbor neuron v, and it shrinks with time. This process is applied for each input vector ζ iteratively.

Neural gas and growing neural gas

The neural gas algorithm is a generalization of the k-means algorithm. The idea of the k-means algorithm is that given a several example vectors, they are clustered into a few classes iteratively, accordingly to a distortion measure (which should be minimized). Every class has its mean vector, and at each iteration every example vector is assigned to the class with the closest mean vector. After that every mean vector is replaced by the average of all vectors in its class.

Then, the difference of the neural gas algorithm is that every example vector is not assigned to a single class but to more than one class. Assignment to the closest class is given with a high weight and to other classes with smaller weights. The mean is updated at each iteration, in the same way as in k-means algorithm.

The update process of weights of neurons in the neural gas algorithm uses two adaptation parameters: α_1 and α_2. The updated formula for a neuron with weight vector at time t is

$$w_{\text{win}(t+1)} = w_{\text{win}}(t) + \alpha_1(t)(\zeta - w_{\text{win}(t)}), \tag{2}$$

$$w_n(t+1) = w_n(t) + \alpha_2(t)(\zeta - w_n(t)). \tag{3}$$

Growing neural gas is a self-organization neural network first proposed by Bernd Fritzke. Unlike the earlier neural gas, growing neural gas can add and delete nodes during algorithm execution. The main idea is to successively add new neurons (units) to an initially small network. The complete algorithm is as follows:

- Start with two units a and b at random positions w_a and w_b in $\mathcal{R}^n$.
- Generate an input signal ζ.
- Find the nearest neuron s_1 and the second nearest unit s_2.
- Increment the age of all edges emanating from s_1.
- Add the squared distance between the input and the nearest neuron in input space to a local counter variable

$$err = err + |w_{s_1} - \zeta|^2.$$

- Move s_1 and its direct topological neighbors toward ζ by fractions α_1 and α_2 respectively of the total distance

$$\Delta w_{s_1} = \alpha_1(\zeta - w_{s_1}), \tag{4}$$

$$\Delta w_n = \alpha_2(\zeta - w_n). \tag{5}$$

- If s_1 and s_2 are connected by an edge, set the age of this edge to zero. If such an edge does not exist, create it.
- Remove edges with an age larger than $c_{\max}$. If this results in points having no emanating edges, remove them as well.
- Insert a new unit as follows:
  - Determine the neuron q with the maximum accumulated error.
  - Insert a new unit r halfway between q and its neighbor f with the largest error variable.
  - Insert edges connecting the new unit r with units q and f and remove the original edge between q and f
  - Decrease the error variables of q and f by multiplying them with a constant. Initialize the error variable of r with the new value of the error variable of q.
- Decrease all error variables by multiplying them with a constant d.
- If a stopping criterion is not yet fulfilled repeat all the previous steps.

Geometric algebra

The geometric or Clifford algebra was introduced by William K. Clifford (1845–1879) and there has been interesting proposals using it in areas as robotics, computer vision, etc. Roughly speaking, geometric algebra (GA) is a mathematical framework which allows us to treat geometric objects as algebraic entities that can be easily treated with algebraic tools. Some authors refer it as *geometric algebra* because they are more interested in geometric interpretation of algebraic entities; while others refer it as *Clifford algebra* because they are more interested in the algebraic aspects. Here we will adopt the term "geometric algebra", and give a brief introduction; interested reader

can see Bayro-Corrochano (2005), Perwass and Hildenbrand (2003), Rosenhahn and Sommer (2002) for more detailed explanations of different geometric algebras.

The geometric algebra $G_{p,q,r}$ is constructed over the vector space $\mathcal{V}^{p,q,r}$, where p, q, r denote the signature of the algebra; if $p \neq 0$ and $q = r = 0$, the metric is Euclidean; if only $r = 0$, the metric is pseudoeuclidean; if $p \neq 0, q \neq 0, r \neq 0$, the metric is degenerate. In this algebra, we have the *geometric product* which is defined as in Equation (6) for two vectors a, b, and have two parts: the inner product $a \cdot b$ is the symmetric part, while the wedge product $a \wedge b$ is the antisymmetric part:

$$ab = a \cdot b + a \wedge b. \tag{6}$$

The dimension of $G_{n=p,q,r}$ is 2^n, and G_n is constructed by the application of the geometric product over the vector basis e_i,

$$e_i e_j = \begin{cases} 1 \text{ for } i = j \in 1, \ldots, p, \\ -1 \text{ for } i = j \in p+1, \ldots, p+q, \\ 0 \text{ for } i = j \in p+q+1, \ldots, p+q+r, \\ e_i \wedge e_j \text{ for } i \neq j. \end{cases}$$

This leads to a basis for the entire algebra: $\{1\}, \{e_i\}, \{e_i \wedge e_j\}, \{e_i \wedge e_j \wedge e_k\}, \ldots, \{e_1 \wedge e_2 \wedge \ldots \wedge e_n\}$. Any multivector can be expressed in terms of this basis. In the $2^n - D$ space there are multivectors of grade 0 (scalars), grade 1 (vectors), grade 2 (bivectors), grade 3 (trivectors). . . up to grade n.

This results in a basis for G_n containing elements of different grade called *blades* (e.g., scalars, vectors, bivectors, trivectors, etc): $1, e_1 \ldots e_{12} \ldots e_{123} \ldots I$, which is called *basis blade*; where the element of maximum grade is the pseudoscalar $I = e_1 \wedge e_2 \ldots \wedge e_n$. A linear combination of basis blades, all of the same grade k, is called k-vector. The linear combination of such k-vectors is called *multivector*, and multivectors with certain characteristics represent different geometric objects (as points, lines, planes, circles, spheres, etc), depending on the GA where we are working in. For example, a point is represented in $G_{3,0,0}$ (the GA of the 3D Euclidean space, $\mathcal{E}^3$) as $x = ae_1 + be_2 + ce_3$, while in $G_{3,1,0}$ (the GA of the projective space), it is represented as $x = ae_1 + be_2 + ce_3 + e_4$; however, a circle cannot be defined in $G_{3,0,0}$ as a single multivector (although it is possible to define it parametrically), but it is possible to define it in $G_{4,1}$ as a 2-vector $Z = S_1 \wedge S_2$ (the intersection of two spheres in the same space).

Given a multivector M, it is called homogeneous if it contains just elements of the same grade. Given a multivector M, if we are interested in extracting only the blades of a given grade, we write $\langle M \rangle_r$ where r is the grade of the blades we want to extract (obtaining an homogeneous multivector M' or a r-vector). The *dual* of M is computed by multiplying the multivector M by the pseudo-scalar of the respective GA: $M^* = IM$.

Rigid-body motions are defined in geometric algebra using entities named *versors*; special versors are the so-called *rotors, translators and motors*, which rotate, translate and apply both transformations, respectively. These versors are applied multiplicatively to the entities to be transformed. However, its important to note that, although rotations are applied in a multiplicative way in $G_{3,0,0}$, translations are applied by direct sum, and to apply it multiplicatively we need to move to other algebra. Such transformations are ideally managed by the conformal geometric algebra $G_{4,1,0}$, which is explained below. Interested reader can see Bayro-Corrochano (2005), Rosenhahn and Sommer (2002) for more details and explanations of different geometric algebras.

Conformal geometric algebra

Conformal geometric algebra (CGA) $G_{4,1,0}$ is applied to embed the Euclidean space in a higher dimensional space with two extra basis vectors which have particular meaning; in this way, we represent particular objects of the Euclidean space with subspaces of the conformal space. The vectors we add are e_+ and e_-, which square to 1, -1, respectively. With these two vectors, we define the null vectors

$$e_0 = \frac{1}{2}(e_- - e_+); \quad e_\infty = (e_- + e_+), \tag{7}$$

interpreted as the origin and the point at infinity, respectively. From now and in rest of the paper, points in the 3D Euclidean space are represented in lowercase letters, while conformal points in uppercase letters; also the conformal entities will be expressed in the *inner product null space* (IPNS), and not in the *outer product null space* unless it is specified explicitly. To map a point $x \in \mathcal{V}^3$ to the conformal space in $G_{4,1}$, we use

$$X = x + \frac{1}{2}x^2 e_\infty + e_0. \tag{8}$$

As mentioned before, we can use CGA to represent particular objects of the 3D Euclidean space; the spheres are specially interesting because they are the basic entities in CGA from which other entities are derived. Spheres with center in c and radius ρ are represented as

$$S = c + \frac{1}{2}(c^2 - \rho^2)e_\infty + e_0. \tag{9}$$

In fact, we can think in conformal points X as degenerated spheres of radius $\rho = 0$.

Let X_1, X_2 be two conformal points. If we subtract X_2 from X_1, we obtain

$$X_1 - X_2 = (x_1 - x_2) + \frac{1}{2}(x_1^2 - x_2^2)e_\infty + e_0 \tag{10}$$

and if we square this result, we obtain

$$(X_1 - X_2)^2 = (x_1 - x_2)^2. \tag{11}$$

So, if we want a measure of the euclidean distance between the two points, we can apply (11). Reader is encouraged to see the CGA representation of other entities from Rosenhahn and Sommer (2002). All of such entities and its transformations can be managed easily using the rigid-body motion operators described further.

Rotation, translation and dilation

In GA there exist specific operators named *versors* to model rotations, translations and dilations, they are called rotors, translators and dilators respectively. In general, a versor G is a multivector which can be expressed as the geometric product of nonsingular vectors

$$G = \pm a_1 a_2 \ldots a_k. \tag{12}$$

In CGA such operators are defined by (13), (14) and (15), being R the rotor, T the translator, and D_λ the dilator,

$$R = e^{\frac{1}{2}\mathbf{b}\theta}, \tag{13}$$

$$T = e^{-\frac{te_\infty}{2}}, \tag{14}$$

$$D_\lambda = e^{\frac{-\log(\lambda)\wedge E}{2}}, \tag{15}$$

where $\mathbf{b}$ is the bivector dual to the rotation axis, θ is the rotation angle, $t \in \mathcal{E}^3$ is the translation vector, λ is the factor of dilation and $E = e_\infty \wedge e_0$.

Such operators are applied to any entity of any dimension by multiplying the entity by the operator from the left, and by the reverse of the operator from the right. Let X_i be any entity in CGA; then to rotate it we apply (16), while to translate it we apply (17), and to dilate we use (18). However, dilations are applied only on the origin, so we must translate the entity X_i to origin, then to dilate it, and finally translate it back to its original position, as expressed by (19),

$$X_1' = R X_1 \tilde{R}, \tag{16}$$

$$X_2' = T X_2 \tilde{T}, \tag{17}$$

$$X_3' = D_\lambda X_3 \tilde{D}_\lambda, \tag{18}$$

$$X_4' = \tilde{T}_o D_\lambda T_o X_4 \tilde{T}_o \tilde{D}_\lambda T_o. \tag{19}$$

Determining the shape of an object

To determine the shape of an object, we can use a topographic mapping which uses selected points of interest along the contour of the object to fit a low-dimensional map to the high-dimensional manifold of such contour. This mapping is commonly achieved by using self-organized neural networks as Kohonen's maps or neural gas (Mehrotra *et al.*, 1997); however, if we desire a better topology preservation, we should not specify the number of neurons of the network *a priori* (as specified for neurons in SOM or NG, together with its neighborhood relations), but allow the network to grow using an incremental training algorithm, as in the case of the growing neural gas (Fritzke, 1995). In this work we follow the idea of growing the neural network and present two approaches (one based on the SOM algorithm, and other bases on the GNG algorithm) to determine the shape of objects by means of applying versors of the CGA, resulting in a model easy to handle in postprocessing stages, for example modeling the dynamic behavior of the object. A scheme of our approach is shown in Figure 1. This representation uses only one base point and a set of versors in the conformal geometric algebra framework (translators T in 2D, motors M in 3D), which moves such point along the contour of the object we are interested in, to determine its shape. It means that the neural network has versors associated to its neurons, and its adaptation algorithm determines the parameters that best fit the input patterns, allowing us to get every point on the contour by interpolation of such versors. This method is a coordinate- free approach.

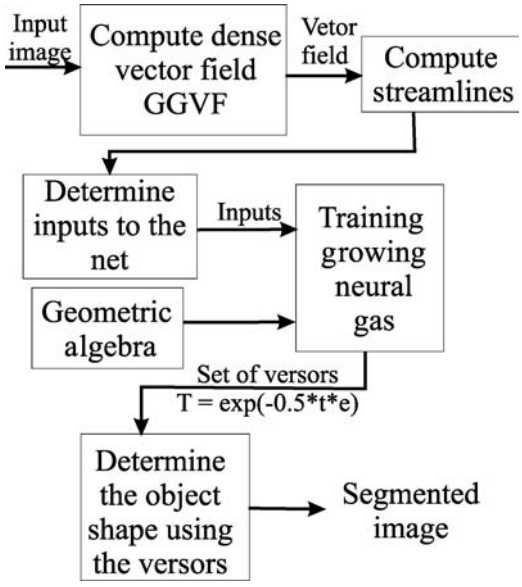

Figure 1 A block diagram of our approach.

In addition, we modify the acquisition of input patterns by adding a preprocessing stage which determines the inputs to the net; this is done by computing the GGVF and analyzing the *streamlines* followed by particles (points) placed on the vertices of small squares by dividing the 2D or 3D space in such squares (or cubes in 3D case). The streamline or the path followed by a particle that is placed on $\mathbf{x} = (x, y, z)$ coordinates will be denoted as $S(\mathbf{x})$. The information obtained with GGVF is also used in the adaptation stage as explained below.

Automatic samples selection using GGVF

Since our goal is to have an approach which needs less intervention of users, the selection of input patterns must be automatic and robust; it means that we want to give to the computer only the medical image or the volumetric data in order to find the shape of the object we are interested in. Therefore, we need a method that can provide information

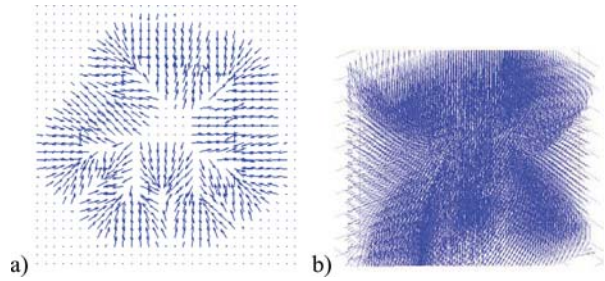

Figure 2 Example of the dense vector field called GGVF (it is shown not all the vector field, but only representative samples of a grid). (a) Samples of the vector field for a 2D image; (b) samples of the vector field for volumetric data.

to guide the algorithm in this selection. The GGVF (Xu, 1999) is a *dense vector field* derived from the volumetric data by minimizing a certain energy functional in a variational framework. The minimization is achieved by solving linear partial differential equations which diffuses the gradient vectors computed from the volumetric data. To define the GGVF, the edge map is defined at first. Let $f(\mathbf{x}) : \Omega \to \mathcal{R}$ be an edge map defined in $\Omega \subset \mathcal{R}^{\mathbf{n}}$; then the GVF in Ω is defined as the vector field $\mathbf{v}(\mathbf{x}) : \Omega \to \mathcal{R}$ which minimizes the energy functional,

$$\mathcal{E} = \int_{\Omega} g(|\nabla f|)\nabla^2 \mathbf{v} - h(|\nabla f|)(\mathbf{v} - \nabla f), \qquad (20)$$

where the gradient operator ∇ is applied to each component of $\mathbf{v}$ separately. For the 2D case, it is defined as $f(x, y) = -|\nabla G(x, y) * I(x, y)|^2$, where $I(x, y)$ is the gray level of the image on pixel (x, y), $G(x, y)$ is a 2D Gaussian function (for robustness in presence of noise), and ∇ is the gradient operator,

$$g(|\nabla f|) = e^{-\frac{|\nabla f|}{\mu}} \quad \text{and} \quad h(|\nabla f|) = 1 - g(|\nabla f|) \quad (21)$$

and μ is a regularization parameter governing the trade-off between the first term and the second term in the integrand. This parameter should be set according to the amount of noise present in the image (more noise, increase μ). An example of such dense vector field obtained in a 2D image is shown in Figure 2(a), while an example of the vector field for a volumetric data is shown in Figure 2(b). Observe the large range of capture of the forces in the image. Due to this large capture range, if we put particles (points) on any place over the image, they can be guided to the contour of the object.

The automatic selection of input patterns is done by analyzing the *streamlines* of points of a 3D grid topology defined over the volumetric data. It means that the algorithm follow the streamlines of each point of the grid, which will guide the point to the more evident contour of the object; then the algorithm selects the point where the streamline finds a peak in the edge map and gets its conformal representation X (as in Equation (8)) to make the inputs pattern set. Additionally, to the X (conformal position of the point), the

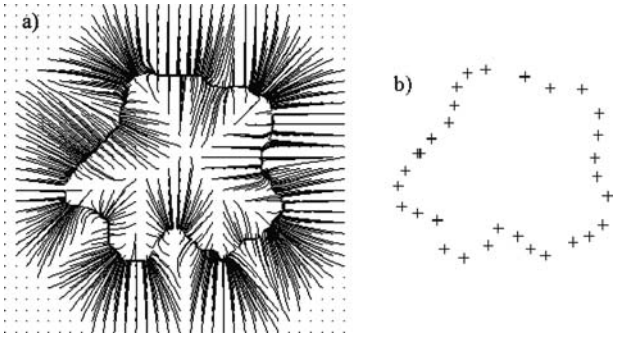

Figure 3 (a) Example of streamlines for particles arranged in a 32×32 grid according to the vector field shown in Figure 2(a); (b) selected points as input patterns according to Equation (22).

inputs have the vector $\mathbf{v}_\zeta = [u,\ v,\ w]$ which is the value of the GGVF in such pixel and it will be used in the training stage as a parameter determining the amount of energy that the input has to attract neurons. This information will be used in the adaptation stage together with the position $\mathbf{x}$ to approximate the topology of the data. Summarizing, the input set $\mathbf{I}$ will be

$$\mathbf{I} = \{\zeta_k = (X_{\zeta_k}, \mathbf{v}_{\zeta_k}) | \mathbf{x}_\zeta \in S(\mathbf{x}')\ and\ f(\mathbf{x}_\zeta) = 1\}, \quad (22)$$

where X_ζ is the conformal representation of $\mathbf{x}_\zeta$; $\mathbf{x}_\zeta \in S(\mathbf{x}')$ means that $\mathbf{x}_\zeta$ is on the path followed by a particle placed in $(\mathbf{x}')$, and $f(\mathbf{x}_\zeta)$ is the value of the edge map in position $\mathbf{x}_\zeta$ (assuming it is binarized). As some streamlines can carry to the same point or very close points, we can add constraints to avoid very close samples; one very simple restriction is that the candidate to be included in the input set must be at least at a fixed distance d_{thresh} of any other input.

Selection of input patterns by this way avoids the necessity of preprocessing the image (thresholding and highlighting contours) and manual selection, as in Angelopoulou *et al.* (2005); this is because the GGVF is very robust even in the presence of noise, and is guaranteed that streamlines will guide us to the contours of the image. Figure 3 shows the streamlines according to the vector field shown in Figure 2(a) and the input patterns selected as described before.

Shape approximation using versors

Using each one of the neural nets mentioned earlier, we will define the versors that applied to a point which will describe the shape of the object. First, we present the algorithm for the *growing SOM* (GSOM) in some detail, then we only specify the changes to adapt it to the GNG approach. It is important to note that, although we are explaining the algorithm using points, the versors can be applied to any entity in GA that we had selected to model the object. The network starts with a minimum number of versors (neural units) and new units are inserted successively. The network is specified by the following:

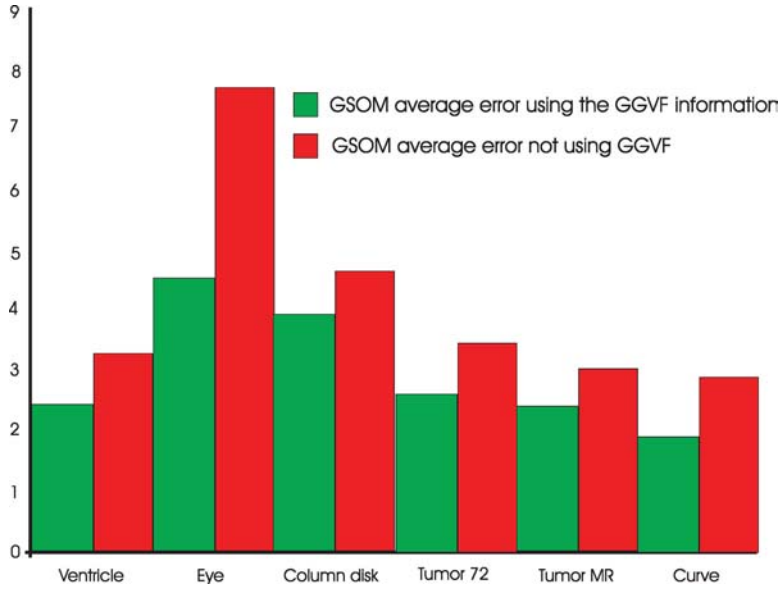

Figure 4 Average errors for different examples using the GSOM algorithm with and without GGVF information.

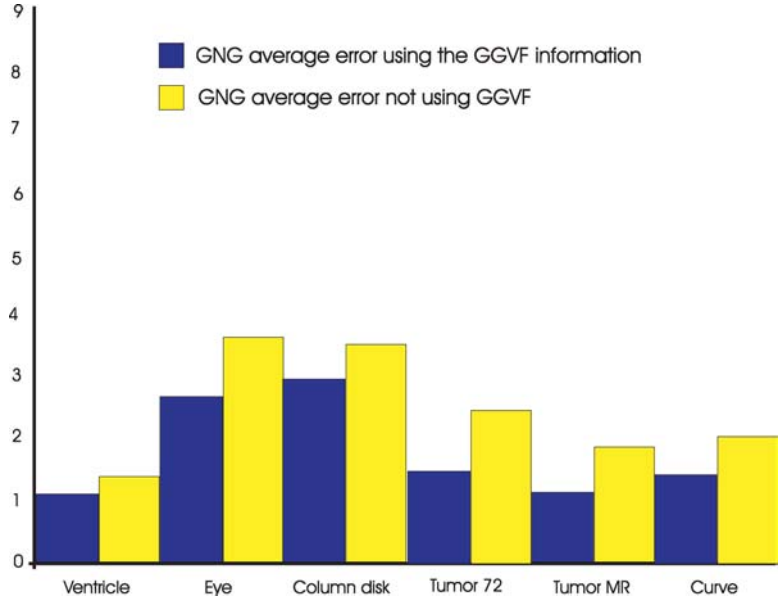

Figure 5 Average errors for different examples using the GNG algorithm with and without GGVF information.

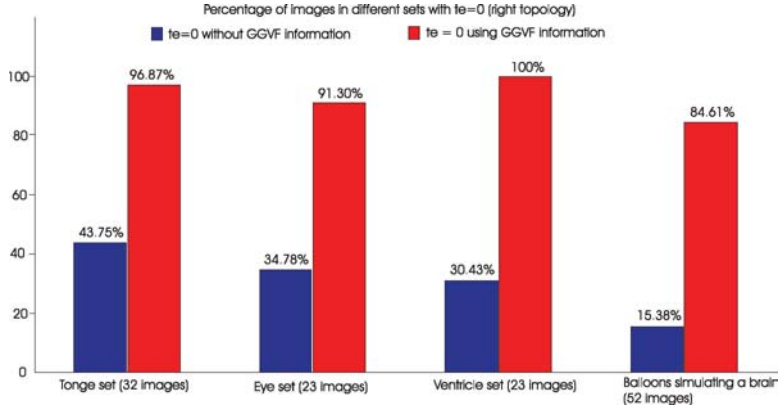

Figure 6 Percentage of images in four sets for which the topographic error te is zero after reaching the stop criterion.

- A set of units (neurons) named N, where each $\mathbf{n}_l \in N$ has its associated versor $M_{\mathbf{n}_l}$; each versor is the transformation that must be applied to a point to place it in the contour of the object. The set of transformations will ultimately describe the shape of the object.
- A set of connections between neurons defining the topological structure.

In this approach, we will use the information available on GGVF to guide the adaptation. With this elements, we define the adaptation algorithm of the growing SOM to find the versors that will define the contour as follows:

1. Let P_0 be a fixed initial point over which the transformations will be applied. Such transformations will be expressed as $M = e^{-\frac{t}{2}e_\infty}$ in the conformal geometric algebra. This point corresponds to the conformal representation of p_0, which can be a random point or the centroid defined by the inputs. The vector t will be determined according the distance between X_ζ and P_0 as explained below, but initially it is a random displacement.
2. Start with the minimal number of neurons, which have associated random translators as well as a vector $\mathbf{v_l} = [u_l,\ v_l,\ w_l]$ whose magnitude is interpreted as the "availability for adaptation" of such neuron (initially set to 1).
3. Select one input ζ from the inputs set $\mathbf{I}$ and find the winner neuron; that means to find the neuron n_l having the versor M_l which moves the point P_0 closer to such input:

$$M_{\text{win}} = \min_{\forall M} \sqrt{(X_\zeta - M P_0 \tilde{M})^2}. \tag{23}$$

4. Modify M_{win} and all others versors of neighboring neurons M_l in such a way that the modified T will represent a transformation moving the point P_0 nearer the input:

$$M_{l_{\text{new}}} = e^{-\frac{t_l}{2}e_\infty} e^{-\frac{\Delta t_l}{2}e_\infty}, \tag{24}$$

where

$$\Delta t_l = \alpha\ \phi\ \eta(\mathbf{v}_\zeta, \mathbf{v_l})\ (\mathbf{x}_\zeta - p_0), \tag{25}$$

α is an adaptation parameter, ϕ is a function defining the amount of a neuron that can learn according to its distance to the winner neuron (usually defined as in (26)), and $\eta(\mathbf{v}_\zeta, \mathbf{v_l})$ is defined as in (27),

$$\phi = e^{-\frac{(M_{\text{win}} P_0 \tilde{M}_{\text{win}} - M_l P_0 \tilde{M}_l)^2}{2\sigma}} \tag{26}$$

$$\eta(\mathbf{v}_\zeta, \mathbf{v_l}) = \|\mathbf{v}_\zeta - \mathbf{v_l}\|^2. \tag{27}$$

which acts as a parameter indicating the quantity of adaptation depending on the teaching strength of the input ζ and the learning capacity of the neuron, given in $\mathbf{v}_\zeta$ and $\mathbf{v_l}$, respectively. In other words, with $\eta(\mathbf{v}_\zeta, \mathbf{v_l})$ we are taking into account the information of GGVF which guide to the contours. Finally, also update the value $\mathbf{v_l}$:

$$\mathbf{v}_{\mathbf{l}_{\text{new}}} = (\mathbf{1} + \alpha\ \ \phi)\ \ \mathbf{v_l} \tag{28}$$

5. Insert new neurons as follows:
   For the 2D case:
   - Determine neighboring neurons $\mathbf{n_i}$ and $\mathbf{n_j}$ connected by an edge larger than $c_{\max}$.
   - Create a new neuron $n_{\max}$ between $\mathbf{n_i}$ and $\mathbf{n_j}$ whose associated M and $\mathbf{v_l}$ will be

$$M_{n_{\text{new}}} = \frac{M_i + M_j}{2}, \tag{29}$$

$$\mathbf{v}_{\mathbf{l}_{\text{new}}} = \frac{\mathbf{v_i} + \mathbf{v_j}}{2} \tag{30}$$

   - Delete old edge connecting $\mathbf{n_i}$ and $\mathbf{n_j}$ and create two new edges connecting n_{new} with $\mathbf{n_i}$ and $\mathbf{n_j}$.

   Note: For the 3D case, we do not present the insertion method because we will not use this method in 3D. This is because, as will be seen in the experiments section, the GNG method (explained in the next section), performs better than this approach in 2D (which is generalizable for 3D).
6. Decrease the adaptation parameter α by a factor r

$$\alpha_\tau = \alpha_{\tau-1} * r \tag{31}$$

   and repeat steps 3 to 5 until reach some predefined value for α, or when reach the maximum number of neurons.

Training the network we find the set of M defining positions on a trajectory; such positions minimizes the *quantization error* measured as the average distance between X_ζ and the result of $M_\zeta P_0 \tilde{M}_\zeta$:

$$\chi = \frac{\sum_{\forall \zeta} (\sqrt{(M_\zeta P_0 \tilde{M}_\zeta - X_\zeta)^2}}{N}, \tag{32}$$

where M_ζ moves P_0 closer to input X_ζ, and N is the number of inputs.

One important thing to note is that, although the quantization error measures the average distance between each input and its best-matching unit, it does not state anything about the topology preservation. The *topographic error* measures the proportion of all inputs, for which the first and second best matching units are not adjacent. The topographic error is defined as

$$te = \frac{1}{N} \sum u\left(\zeta_{i\,i=1}^{N}\right). \tag{33}$$

GNG using GGVF

For the case of the growing neural gas, the adaptation algorithm is very similar to the one explained for the GSOM. Actually, we only have to take into account that

- There are two adaptation parameters: e_w and e_n, for the winner neuron and for the direct neighbors of it. These parameters do not decrease with time as the parameter α in the GSOM case, but remain constant during all the

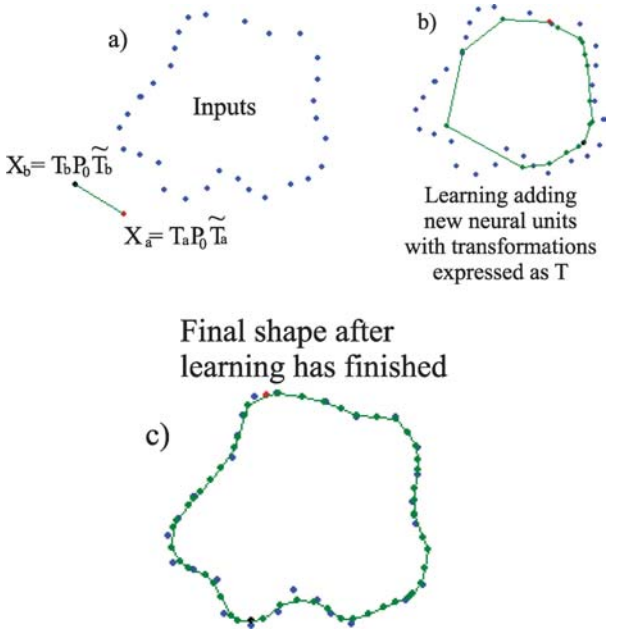

Figure 7 (a) Inputs to the net and the initial shape (a line) defined according the two initial random translators T_a, T_b; (b) shape after only one iteration; c) Final shape defined with the 50 estimated translators.

process. This implies that to update the transformation M we use

$$\Delta t_{\text{win}} = e_w \, \eta(\mathbf{v}_\zeta, \mathbf{v}_{\mathbf{win}}\mathbf{a}) \, (\mathbf{x}_\zeta - p_0) \tag{34}$$

for the winner neuron, and

$$\Delta t_n = e_n \, \eta(\mathbf{v}_\zeta, \mathbf{v}_\mathbf{n}) \, (\mathbf{x}_\zeta - p_0) \tag{35}$$

for its direct neighbors. Also update

$$\mathbf{v}_{\mathbf{win}}^{\mathbf{new}} = (\mathbf{1} + \alpha \;\; \phi) \;\; \mathbf{v}_{\mathbf{win}}^{\mathbf{new}} \tag{36}$$

$$\mathbf{v}_{\mathbf{n}}^{\mathbf{new}} = (\mathbf{1} + \alpha \;\; \phi) \;\; \mathbf{v}_{\mathbf{n}.}^{\mathbf{new}} \tag{37}$$

- Each neuron $\mathbf{n_l}$ will be composed of its versor $M_{\mathbf{n_l}}$ and two new attributes: the *signal counter* sc_l and the *relative signal frequency* rsf_l. The signal counter sc_l is incremented for the neuron $\mathbf{n_l}$ every time it is the winner neuron. The relative signal frequency rsf_l is defined as

$$rsf_l = \frac{sc_l}{\sum_{\forall \mathbf{n_j}} sc_j}. \tag{38}$$

  This parameter will act as an indicator to insert new neural units.
- The process to insert new neural units changes. Here we take into account the parameter rsf_l in the following way: every certain number λ of iterations determine the neuron with the rsf_l with highest value. Then, if any of the direct neighbors of that neuron is at a distance larger than $c_{\max}$, insert a new neuron in the same way as in step 5 of the GSOM algorithm, using Equation (30).
- When inserting new neurons, the new units will have the values $sc_{\text{new}} = 0$ and $rsf_{\text{new}} = 0$.

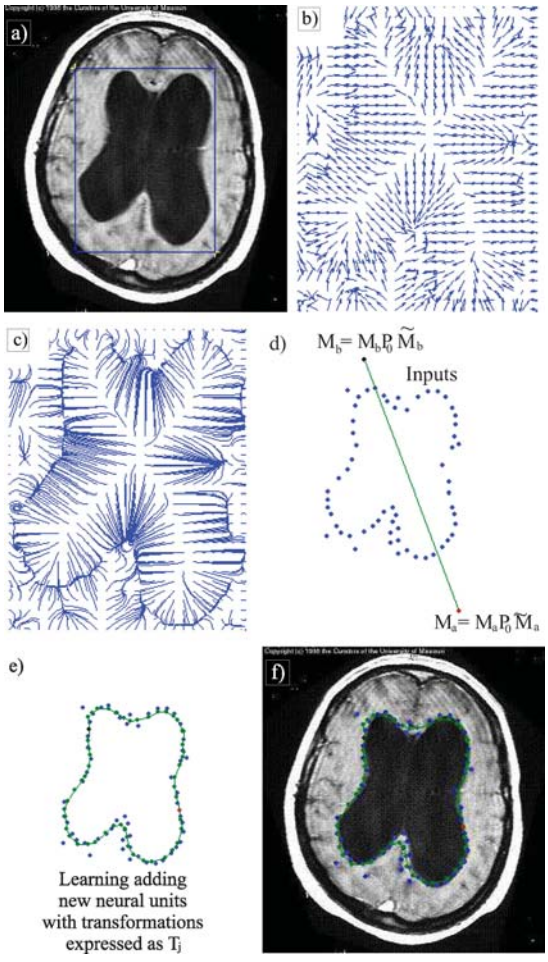

Figure 8 (a) Original image and region of interest (ROI); (b) zoom of the dense vector field of the ROI; (c) zoom of the streamlines in ROI; (d) inputs and initial shape; (e) final shape defined according the 54 estimated translators; (f) image segmented according the results.

- The stop criterion is when a maximum number of neurons is reached or when the parameter called "availability for adaptation" of neurons approaches to zero (i.e., it is less that a threshold $c_{\min} \in \mathcal{R}$), the first that happens will stop the adaptation process.

The errors with this approach are measured using (32) and (33).

Experiments

Both algorithms were applied to a set of 2D medical images (some obtained with computer tomography (CT) and some with magnetic resonance (MR)). Figure 4 shows the average errors using the GSOM algorithm with different examples: segmenting a ventricle, a blurred object, a free-form curve, and a column disk. For each example (image), the error obtained with GSOM using the GGVF information and without it is showed as a bar. Note that using the GGVF information the error is reduced. This

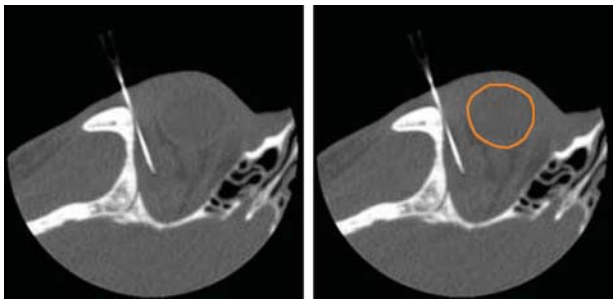

Figure 9 (a) Original image containing a circular "object", with very blurred contours; (b) ground-truth of the object we are interested in (marked by hand).

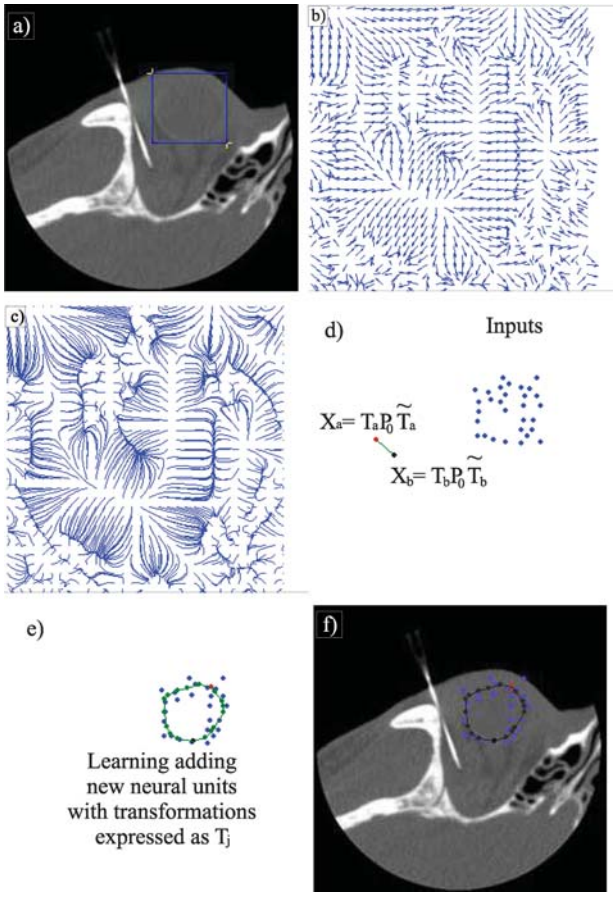

Figure 10 (a) Original brain image and the region of interest (ROI); (b) zoom of the dense vector field of the ROI; (c) zoom of the streamlines in ROI; (d) inputs and initial shape according the two initial random transformations T_a and T_b; (e) final shape defined according the 25 estimated translators; (f) original image with the segmented object.

means that using the GGVF information, as we propose, a better approximation of the object shape can be obtained. Figure 5 shows the average errors obtained for the same examples, but using the GNG with and without the GGVF information. Note that the GGVF contributes to obtain a better approximation to the object surface. Also note that the average errors obtained with the GNG algorithm

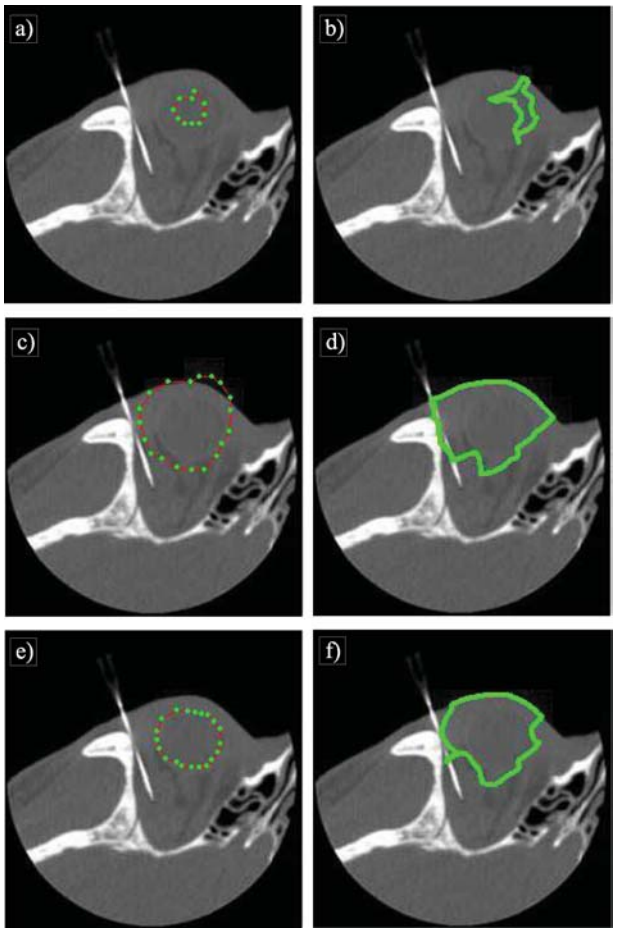

Figure 11 Result obtained when using the active contour approach to segment the object in the same brain image as in Figure 10. Note that, although such approach uses GGVF information, it fails to segment the object no matter if the initialization of the snake is given inside, outside or over object's contour. (a) Initialization of snake inside the object; (b) final result obtained with initialization showed in (a); (c) initialization of snake outside the object; (d) final result obtained with initialization showed in (c); (e) initialization of snake over the contour; (f) final result obtained with initialization showed in (e).

are smaller than the errors obtained with the GSOM. In conclusion, both GSOM and GNG, are improved with GGVF information, but GNG with GGVF gives the best results; for this reason, all the images of the experiments showed in this section correspond to the applications of this approach. If we measure the topographic error (33), we see that the GNG with GGVF information preserves the topology better than the same approach without such information. Several experiments were carried out using GNG with and without the GGVF; Figure 6 shows the percentage of images in four sets for which the topographic error te is zero after reaching the stop criterion.

To illustrate the algorithm explained in "CNG using GGVF" section, we first present the process for the image of the curve shown in Figure 2(a). The inputs to the net

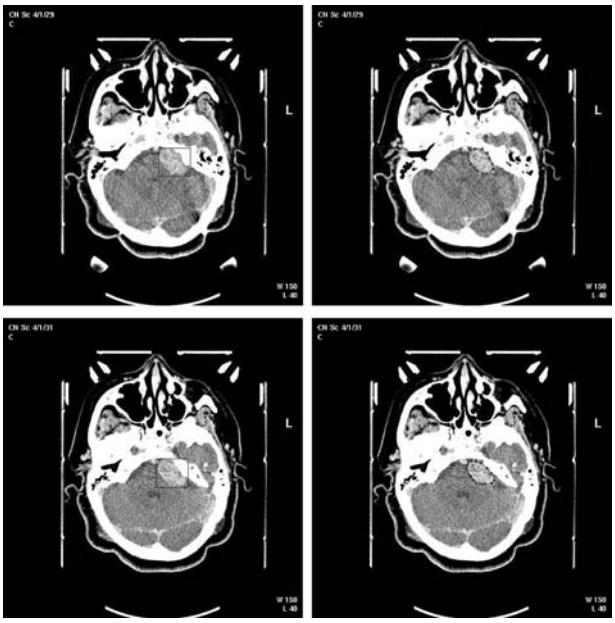

Figure 12 Left column: original image and the region of interest. Right column: Result of segmentation.

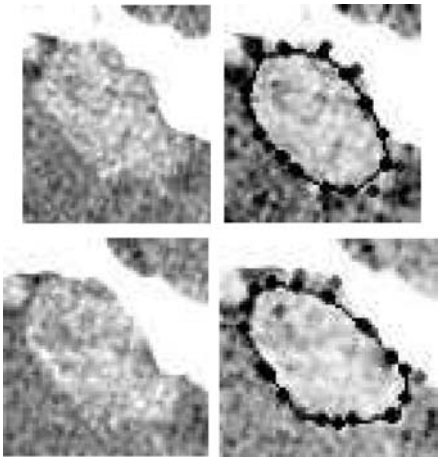

Figure 13 Left column: zoom of the original region of interest. Right column: zoom of the result.

were selected according the process described in "automatic samples selection using GGVF"; that is, computing the vector field GGVF (Fig. 2b) and the streamlines (Fig. 3a), resulting in the input set shown in Figure 3(b). Figure 7' shows different stages in the development of the algorithm described before; Figure x7(a) shows the two initial positions resulting from the application of T_a and T_b (the translators associated with the first two neurons) to P_0; Figure 7(b) shows the result after only one iteration of the algorithm; Figure 7(c) shows the final shape the when net stops.

Figure 8 shows the result when algorithm is applied to a magnetic resonance image (MRI); the goal is to obtain the shape of the ventricle. Figure 8(a) shows the original brain image and the region of interest (ROI); Figure 8(b) shows the computed vector field for the ROI; Figure 8(c) shows

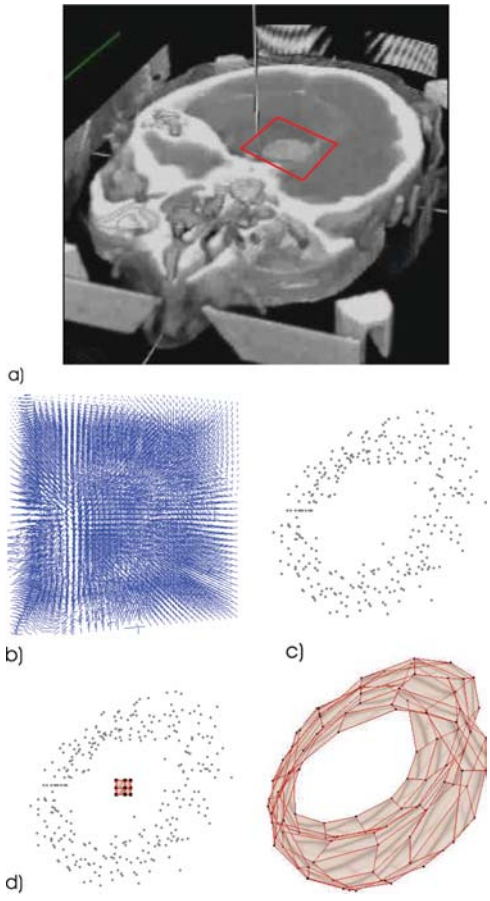

Figure 14 The algorithm for determining the shape of 3D object. (a) 3D model of the patient's head containing a tumor in the marked region; (b) vectors of the dense GGVF on a 3D grid arrangement of $32 \times 32 \times 16$; (c) inputs determined by GGVF and edge map; (d) inputs and the initialization of the net GNG; (e) final shape after training has finished with a total of 300 versors M (associated with 300 neural units).

the streamlines in the ROI defined for particles placed on the vertices of a 32×32 grid; Figure 8(d) shows the initial shape as defined for the two initial random translators T_a, T_b; Figure 8(e) shows the final shape obtained; and finally Figure 8(f) shows the original image with the segmented object.

Figure 9 shows a computer tomography image which has a very blurred object: Figure 9(a) is the original image and 9(b) is the ground-truth of the contour. We select such image to show that even in very blurred or noisy images, the algorithm gives good enough results. In fact, if the GGVF-Snake (Xu, 1999) (for which the GGVF theory was developed) is used, it fails to find the shape of the object no matter if the initialization of the snake is given inside, outside or over the (blurred) contour of the object (as shown in Fig. 11); while using the translators T learned with the proposed GNG algorithm, the resulting shape is closer to the expected shape. Figure 10 shows the process for the CT image; note that we only work in a region of interest. Figure 10(a) shows the original image and the

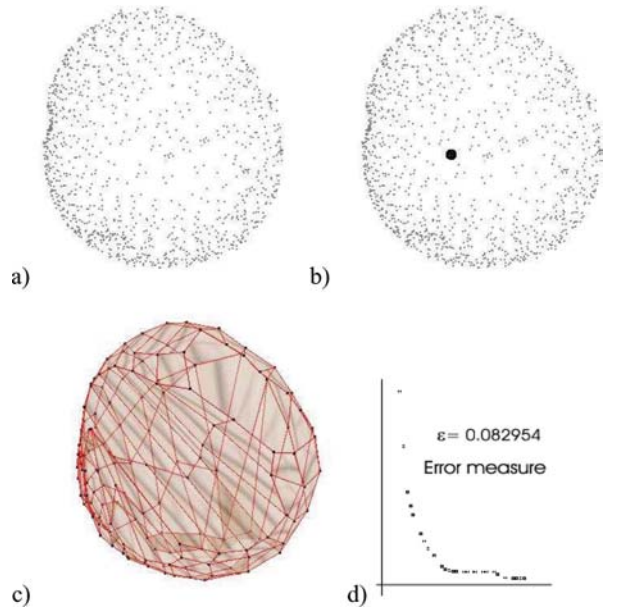

Figure 15 Examples of 3D object shape definition. (a) Inputs to the net selected using GGVF and streamlines; (b) inputs and the initialization of the net with nine neural units; (c) result after the net has been reached the maximum number of neurons (300 neurons); (d) error measurement using Equation (32).

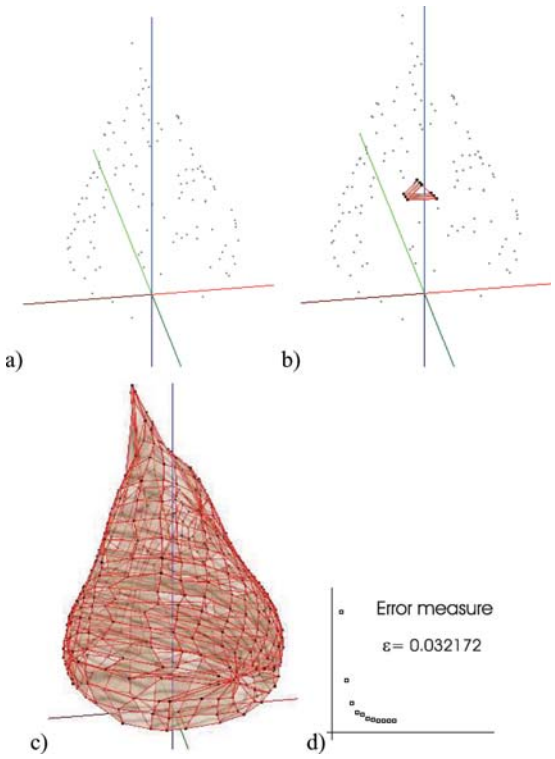

Figure 16 3D object shape definition for the case of a pear. (a) Inputs to the net selected using GGVF and streamlines; (b) inputs and the initialization of the net with nine neural units; (c) result after the net has been reached the maximum number of neurons (300 neurons); (d) error measurement using Equation (32).

region of interest (ROI); Figure 10(b) shows the computed vector field of the ROI; Figure 10(c) shows the streamlines defined for particles placed on the vertices of a 32×32 grid; Figure 10(d) shows the inputs selected according the streamlines and the initial shape as defined for the two initial random translators T_a, T_b; Figure 10(e) shows the final shape obtained; and finally Figure 10(f) shows the result overlapped with the original image, showing that the algorithm gives good results if it is used for segmentation.

It is important to note that, although the approaches of Figures 10 and 11 use GGVF information to find the shape of an object, the estimated final shape is better using the neural approach than the one using active contours; the second approach (see Fig. 11) fails to segment the object no matter if the initialization of the snake is given inside, outside or over the contour we are interested in. Additionally, the fact of expressing such shape as a set of translators allows us to have a model best suited to be used in further applications which can require the deformation of the model, specially if such model is not based on points but on other GA entities, because we do not need to change the translators (remember that they are applied in the same way to any other entity).

The proposed algorithm was applied to different sets of medical images. Figure 12 shows some CT images of a patient with a tumor. The left column shows the original image and the region of interest, while the right column shows the result of the proposed approach. Figure 13 shows a zoom of the region of interest for better visualization.

For the 3D case, Figure 14(a) shows the patient head with the tumor which surface we need to approximate; Figure 14(b) shows the vectors of the dense GGVF on a 3D grid arrangement of size $32 \times 32 \times 16$; Figure 14(c) shows the inputs determined by GGVF and edge map; Figure 14(d) shows the initialization of the net GNG; Figure 14(e) shows final shape after training has finished with a total of 300 versors M (associated with 300 neural units).

Figures 15 and 16 and 16 show the results obtained with other examples using volumetric data, note that the last one corresponds to a pear, and the surface is well approximated. Each figure shows the inputs to the net in (a) selected according the procedure from the section "automatic sample selection using GGVF"; the inputs and the initialization of the net with nine neural units are showed in (b) (the topology of the net is defined as a sort of pyramid around the centroid of input points); while in (c) each figure shows the result after the net has been reached the maximum number of neurons; finally, in (d) each figure shows the minimization of the error according to Equation (32).

It is necessary to mention that the whole process is quick enough; in fact, the computational time required for all the examples showed in this work took only few seconds. The computation of the GGVF is the most time-consuming

task in the algorithm, but it only takes about 3 seconds for 64 $\times$ 64 images, 20 seconds for 256 $\times$ 256 images, and 110 seconds for 512 $\times$ 512 images. This is the reason we decide not to compute it for the whole image, but for selected region of interest. The same criterion was applied to 3D examples.

Conclusions

In this work it was shown the use of the dense vector field named "generalized gradient vector flow" not only to select the inputs to a neural network, but also as a parameter to guide during the adaptation process of the net. The neural networks presented here were the growing Kohonen's map (or growing self-organizing map) and the growing neural gas, which were used to find a set of transformations expressed in the conformal geometric algebra framework, which move a point in a coordinate-free manner by means of a versor along the contour of an object, defining the shape of the object by this way. This is useful because, although we have shown examples using points, the versors of the conformal geometric algebra can be used to transform any entity exactly in the same way: multiplying the entity from the left by M and from the right by $\tilde{M}$. There were presented some experiments and results showing that by incorporating the GGVF information, we can get automatically the set of inputs to the net, and also we improve its performance. It was shown that, although both neural nets are improved, the GNG with GGVF gives superior results when compared with GSOM.

Acknowledgment

The authors would like to thank (Proyecto No. 49) Fondo Sectorial de Salud, SEP-CONACYT and CINVESTAV for supporting this work.

References

Andrade MC. 2004. An interactive algorithm for image smoothing and segmentation. *Electron Lett Comput Vis Image Anal*, 4(1), 32–48.

Angelopoulou A, Psarrou A, García Rodríguez J., Revett K, 2005. 'Automatic landmarking of 2D medical shapes using the growing neural gas network. In Proceedings of the International Conference on Computer Vision, ICCV 2005, October 13–21, Beijing, China, pp. 210–219.

Bayro-Corrochano E. 2005. Robot perception and action using conformal geometry. In *Handbook of Geometric Computing. Applications in Pattern Recognition, Computer Vision, Neurocomputing and Robotics*. E. Bayro-Corrochano (Ed.), Springer, Heidelberg, chap. 13, pp. 405–458.

Fritzke B. 1995. A growing neural gas network learns topologies. *Advances in Neural Information Processing Systems 7*, MIT Press, Cambridge, MA.

Mehrotra K, Mohan C, Ranka S. 1997. Unsupervised learning. *Elements of Artificial Neural Networks*, chap. 5, pp. 157–213.

Moon N, Bullit E, van Leemput K, Gerig G. 2002a.Model based brain and tumor segmentation. In Proceedings of the 16th International Conference on Pattern Recognition, Quebec, Canada, pp. 528–531.

Moon N, Bullit E, van Leemput K, Gerig G. 2002b. Automatic brain and tumor segmentation. In Proceedings of the Fifth International Conference on Medical Image Computing and Computer Assisted Intervention, Tokyo, Japan, pp. 372–379.

Prastawa M, Bullit E, Gerig G. 2003.Robust estimation for brain tumor segmentation. In Conference on Medical Image Computing and Computer Assisted Intervention, vol. 2, pp. 530–537.

Perwass C, Hildenbrand D. 2003. Aspects of geometric algebra in Euclidean, projective and conformal space. Christian-Albrechts-University of Kiel, Technical Report No. 0310.

Rosenhahn B, Sommer G. 2002. Pose estimation in conformal geometric algebra. Christian-Albrechts-University of Kiel, Technical Report No. 0206, pp. 13–36.

Xu Ch. 1999. Deformable models with applications to human cerebral cortex reconstruction from magnetic resonance images. Ph.D. Thesis, Johns Hopkins University, pp. 14–63.

