Object Segmentation Using Growing Neural Gas and Generalized Gradient Vector Flow in the Geometric Algebra Framework

Jorge Rivera-Rovelo[1], Silena Herold[2], and Eduardo Bayro-Corrochano[1]

[1] CINVESTAV Unidad Guadalajara,
Av. Científica 1145, El Bajío, Zapopan, Jalisco, México
{rivera, edb}@gdl.cinvestav.mx
[2] Universidad de Oriente, Santiago de Cuba
silena@csd.uo.edu.cu

Abstract. In this paper we present a method based on self-organizing neural networks to extract the shape of a 2D or 3D object using a set of transformations expressed as versors in the conformal geometric algebra framework. Such transformations, when applied to any geometric entity of this geometric algebra, define the shape of the object. This approach was tested with several images, but here we show its utility first using a 2D magnetic resonance image to segment the ventricle. Then we present some examples of an application for the case of 3D objects.

1 Introduction

The use of neural networks in medical image processing is an area receiving a lot of attention with a variety of applications like segmentation or classification of tissues, etc. The self-organizing neural networks like Kohonen's Map or Self-Organizing Map (SOM), Neural Gas (NG) and Growing Neural Gas (GNG, [3]) have been used broadly when we need to preserve the topology of the data.

In this work we present an approach which uses the *Generalized Gradient Vector Flow (GGVF)* [4] to guide the automatic selection of the input patterns, as well as the learning process of the self-organized neural network GNG to obtain a set of transformations expressed in the conformal geometric algebra framework, which is a coordinate free approach. These transformations help us to define the shape of the object we are interested in. We decided to use such framework because of its coordinate free nature and because it has the advantage that (rigid body) transformations of geometric entities (like points, lines, planes, circles, spheres) are expressed in compact form as operators called *versors*, which are applied in a multiplicatively way to any entity of the conformal geometric algebra. Thus, training the network we do not obtain specific positions for a particular entity (for example, 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.

J.F. Martínez-Trinidad et al. (Eds.): CIARP 2006, LNCS 4225, pp. 306–315, 2006.

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 the geometric visual processing in biological creatures. The experimental results show that approach is very promising.

2 Geometric Algebra

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 pseudo euclidean; 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 (1) 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.

$$ab = a \cdot b + a \wedge b. \tag{1}$$

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.

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

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 n-D space there are multivectors of grade 0 (scalars), grade 1 (vectors), grade 2 (bivectors), grade 3 (trivectors)... up to grade n.

To work in Conformal Geometric Algebra (CGA) $G_{4,1,0}$ means 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

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

interpreted as the origin and the point at infinity, respectively. From now and in the 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

$$X = x + \frac{1}{2}x^2 e_\infty + e_0. \tag{3}$$

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

$$X_1 - X_2 = (x_1 - x_2) + \frac{1}{2}(x_1^2 - x_2^2)e_\infty + e_0 \tag{4}$$

and if we square this result, we obtain

$$(X_1 - X_2)^2 = (x_1 - x_2)^2 \tag{5}$$

So, if we want a measure of the euclidean distance between the two points, we can apply (5). Reader is encouraged to see the CGA representation of other entities consulting [2]. All of such entities and its transformations can be managed easily using the rigid body motion operators described further.

2.1 Rotation, Translation and Dilation

In GA there exist specific operators named *versors* to model rotations, translations and dilations, and 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

$$G = \pm a_1 a_2 ... a_k \tag{6}$$

In CGA, such operators are defined by (7) being R the rotor, T the translator, and D_λ the dilator.

$$R = e^{\frac{1}{2}\mathbf{b}\theta}, \qquad T = e^{-\frac{te_\infty}{2}}, \qquad D_\lambda = e^{\frac{-\log(\lambda)\wedge E}{2}}, \tag{7}$$

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 $E = e \wedge e_0$.

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 be X_i any entity in CGA; then to rotate it, we do $X_1' = RX_1\tilde{R}$, while to translate it we apply $X_2' = TX_2\tilde{T}$, and to dilate we use $X_3' = D_\lambda X_3 \tilde{D}_\lambda$. However, dilations are applied only on the origin, so we must translate the entity X_i to origin, then to dilate it, and finally back translate to its original position.

3 Determining the Shape of an Object

If we want 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 (SOM) or Neural Gas (NG); 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,

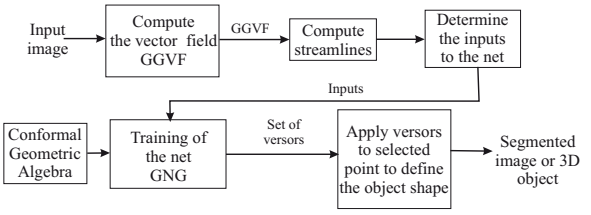

Fig. 1. A block diagram of our approach

as in the case of the Growing Neural Gas (GNG) [3]. In this work we follow the idea of GNG and present an approach to determine the shape of objects by means of applying versors of the CGA, resulting in a model easy to handle in post processing stages, for example modeling the dynamic behavior of the object; a scheme of our approach is shown in Fig. 1. This representation is compact because it 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. That means that the neural network has versors associated to its neurons, and its learning algorithm determines the parameters of them that best fit the input patterns, allowing us to get every point on the contour by interpolation of such versors.

Additionally, 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 Generalized Gradient Vector Flow (GGVF) and analyzing the *streamlines* followed by particles (points) placed on the vertices of small squares by dividing the 3D space in such squares (a sort of grid for 3D). The streamline or the path followed by a particle that is placed on $\mathbf{x} = (x, y, z)$ coordinates will be denoted as $S(\mathbf{x})$.

3.1 Automatic Samples Selection Using GGVF

Since our goal is to have an approach which needs less as possible the intervention of users, the selection of input patterns must be as automatic and robust as possible; that 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 to guide the algorithm in this selection. The GGVF [4] 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 as

$$f(\mathbf{x}) : \Omega \rightarrow \mathcal{R} \tag{8}$$

(for the 2D image, 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).

With this edge map, the GGVF is defined as to be the vector field $\mathbf{v}(x, y, z) = [u(x, y, z), v(x, y, z), w(x, y, z)]$ that minimizes the energy functional

$$\mathcal{E} = \int\int g(|\nabla f|)\nabla^2 \mathbf{v} - h(|\nabla f|)(\mathbf{v} - \nabla f) \tag{9}$$

where

$$g(|\nabla f|) = e^{-\frac{|\nabla f|}{\mu}} \quad \text{and} \quad h(|\nabla f|) = 1 - g(|\nabla f|) \tag{10}$$

and μ is a coefficient. An example of such dense vector field obtained in a 2D image is shown in Fig. 2.a, while an example of the vector field for a volumetric data is shown in Fig. 2.b.

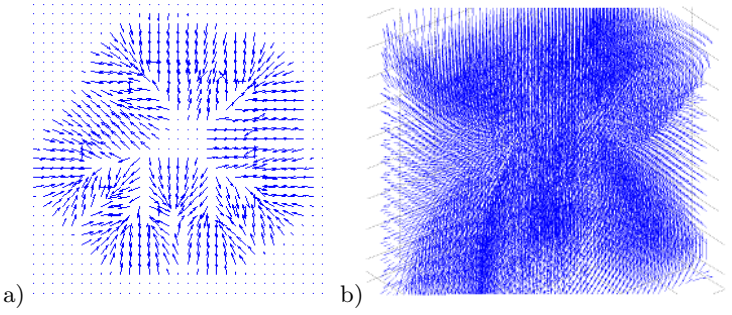

Fig. 2. Example of the dense vector field called GGVF (it is shown not all the vector field, but only representative samples of a grid). a) Samples of the vector field for a 2D image; b) Samples of the vector field for volumetric data.

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 (3))to make the inputs pattern set. Additionally to the X (conformal position of the point), the 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 the input has to attract neurons. This information will be used in the training stage together with the position $\mathbf{x}$ for learning the topology of the data. Summarizing, the input set $\mathbf{I}$ will be

$$\mathbf{I} = \{\zeta_k = (X_{\zeta_k}, \mathbf{v}_{\zeta_\mathbf{k}}) | \mathbf{x}_\zeta \in S(\mathbf{x}')\ and\ f(\mathbf{x}_\zeta) = 1\} \tag{11}$$

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.

3.2 Learning the Shape Using Versors

Using Growing Neural Gas (GNG) we will define the versors that applied to a point will describe the shape of the object. 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 (e.g. planes to describe a surface in 3D, spheres, etc). The network starts with a minimum number of versors (neural units) and new units are inserted successively. The network is specified by

- 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 learning. With this elements, we define the learning algorithm 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 capacity of learning for 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:
$$M_{win} = \min_{\forall M} \|X_\zeta - P_0\| \tag{12}$$
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:
$$M_{l_{new}} = e^{-\frac{t_l}{2}e_\infty} e^{-\frac{\Delta t_l}{2}e_\infty} \tag{13}$$
where
$$\Delta t_l = \alpha\ \phi\ \eta\ (\mathbf{v}_\zeta, \mathbf{v_l})(\mathbf{x}_\zeta - p_0) \tag{14}$$
α is a learning parameter, ϕ is a function defining the amount of a neuron can learn according to its distance to the winner one (usually defined as in (15)), and $\eta(\mathbf{v}_\zeta, \mathbf{v_l})$ is defined as in (16).

$$\phi = e^{-(\frac{X_\zeta - TP_0\tilde{T})^2}{2\sigma}} \tag{15}$$

$$\eta(\mathbf{v}_\zeta, \mathbf{v_l}) = \|\mathbf{v}_\zeta - \mathbf{v_l}\|^2 \tag{16}$$

which is a function defining a quantity of learning depending on the strength to teach of the input ζ and the capacity to learn 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 $\mathbf{v_l}$:

$$\mathbf{v_l} = [(u_l + \alpha\ \phi\ u_l),\quad (v_l + \alpha\ \phi\ v_l),\quad (w_l + \alpha\ \phi\ w_l)]^T \tag{17}$$

5. Insert new neurons as follows:
   - Determine neighboring neurons $\mathbf{n_i}$ and $\mathbf{n_j}$ connected by an edge larger than c_{max}
   - Create a new neuron n_{new} between $\mathbf{n_i}$ and $\mathbf{n_j}$ whose associated M and $\mathbf{v_l}$ will be
   $$M_{n_{new}} = \frac{M_i + M_j}{2}, \qquad \mathbf{v_{l\_new}} = \frac{\mathbf{v_i} + \mathbf{v_j}}{2} \tag{18}$$
   - Delete old edge connecting $\mathbf{n_i}$ and $\mathbf{n_j}$ and create two new edges connecting n_{new} with $\mathbf{n_i}$ and $\mathbf{n_j}$
6. Repeat steps 3 to 5 if the stopping criterion is not achieved. The stopping criterion is when a maximum number of neurons is reached or when the learning capacity of neurons approaches to zero (is less that a threshold c_{min}), the first that happens will stop the learning process.

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

$$\chi = \frac{\sum_{\forall \zeta}(\sqrt{(M_\zeta P_0 \tilde{M}_\zeta - X_\zeta)^2}}{N} \tag{19}$$

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

4 Experiments

To illustrate the algorithm explained in section 3.2 we first present the process for a 2D image. Fig. 3 shows the result when the algorithm is applied to a magnetic resonance image (MRI); the goal is to obtain the shape of the ventricle. Fig. 3.a shows the original brain image and the Region Of Interest (ROI); Fig. 3.b the computed vector field for the ROI; Fig. 3.c the streamlines in the ROI defined for particles placed on the vertices of a 32x32 grid; Fig. 3.d shows the initial shape as defined for the two initial random versors M_a, M_b, which move the point P_0 to $X_a = M_a\ P_0\ \tilde{M}_a$ and $X_b = M_b\ P_0\ \tilde{M}_b$; Fig. 3.e shows the final shape obtained; and finally Fig. 3.f the original image with the segmented object. Many other experiments were carried out, and table 1 shows some quantitative results. Such table shows the errors obtained with the algorithm using and not using the

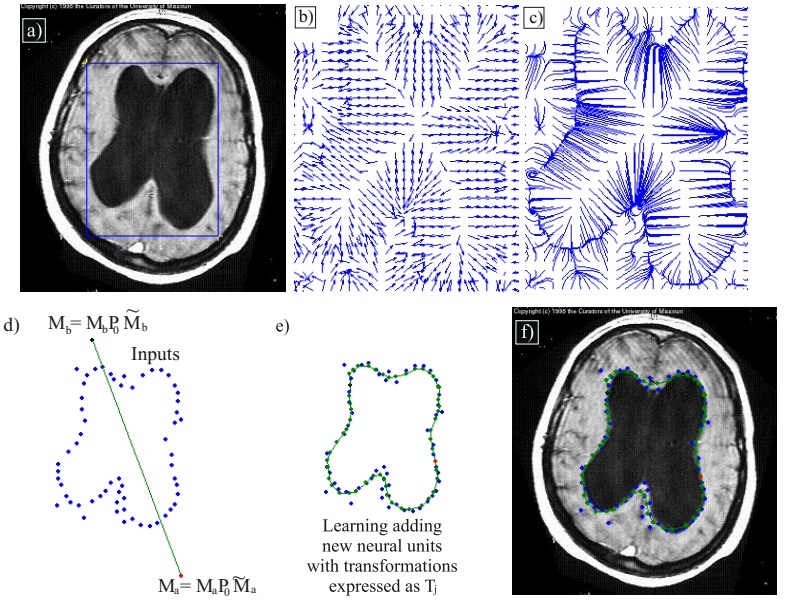

Fig. 3. a) Original image and region of interest (ROI); b) Zoom of the dense vector field of the ROI; c) Zoom of the streamlines in ROI; d) Inputs and initial shape; e) Final shape defined according the 54 estimated translators; f) Image segmented according the results

Table 1. Errors obtained by the algorithm with and without the GGVF information

Example	Error without GGVF	Error with GGVF
Ventricle 1	3.29	2.51
Eye 1	7.63	6.8
Eye 2	3.43	2.98
Column disk 1	4.65	4.1
Tumor 1	3.41	2.85
Tumor 2	2.95	2.41
Free form curve	2.84	1.97
Ventricle 1	3.29	2.51

GGVF information (error measured as in (19)). For the 3D case, Fig. 4.a shows the vectors of the dense GGVF on a 3D grid arrangement of size 32x32x16; Fig. 4.b shows the inputs determined by GGVF and edge map, and also shows the initialization of the net GNG; Fig. 4.c shows Final shape after training has finished with a total of 300 versors M (associated with 300 neural units).

Figure 5 shows the results obtained with other two examples using volumetric data. The first column shows the inputs to the net, selected according the procedure of Sect. 3.1 and the initialization of the net with nine neural units (the topology of the net is defined as a sort of pyramid around the centroid of input

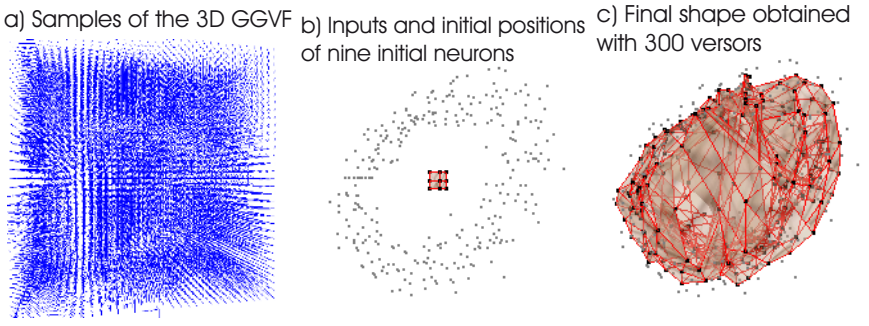

Fig. 4. The algorithm for 3D object's shape determination. a) Vectors of the dense GGVF on a 3D grid arrangement of 32x32x16; b) Inputs determined by GGVF and edge map and the initialization of the net GNG; c) Final shape after training has finished with a total of 300 versors M (associated with 300 neural units).

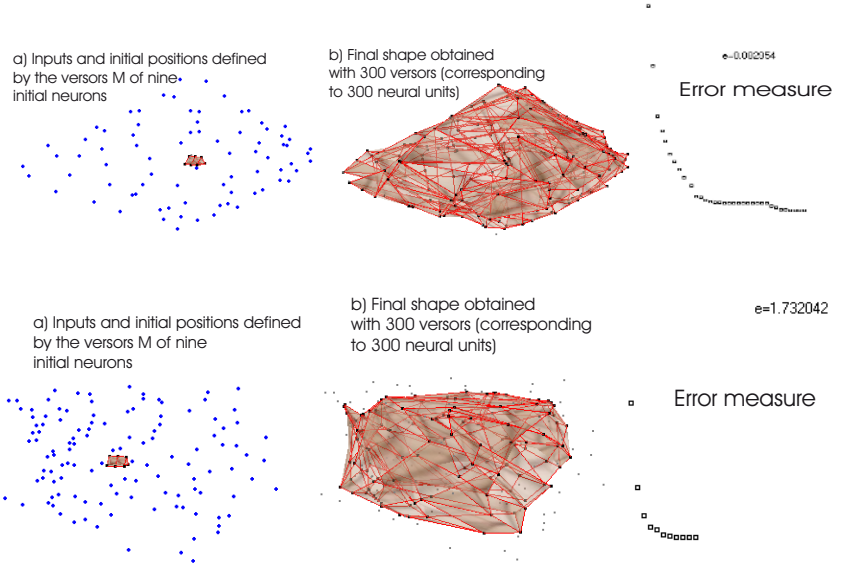

Fig. 5. Two examples of 3D object shape definition. **First column:** inputs to the net selected using GGVF and streamlines, and the initialization of the net with nine neural units; **Second column:** result after the net has been reached the maximum number of neurons (300 neurons); **Third column:** error minimization according to (19).

points); the second column show the result after the net has been reached the maximum number of neurons, which was fixed to 300; the third column shows the minimization of the error according to (19).

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 64x64 images, 20 seconds for 256x256 images, and 110 seconds for 512x512 images. This is the reason why we decide do not compute it for the whole image, but for selected region of interest. The same criterion was applied to 3D examples.

5 Conclusions

In this work it was shown the use of the dense vector field named Generalized Gradient Vector Flow (GGVF) not only to select the inputs to a neural network, but also as a parameter guiding the learning process of the net. The neural network presented here is the growing Neural Gas, which is used to find a set of transformations expressed in the conformal geometric algebra framework, which move a point by means of a versor along the contour of an object, defining by this way the shape of the object. 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 shows that in addition to the set of versors available even if the entity used is other than points, this algorithm is well suited for segmentation tasks.

References

1. E. Bayro-Corrochano, "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 Verlag, Heidelberg, 2005, chap. 13, pp. 405-458.
2. B. Rosenhahn, and G. Sommer, "Pose Estimation in Conformal Geometric Algebra", Christian-Albrechts-University of Kiel, Technical Report No. 0206, pp. 13-36, 2002.
3. B. Fritzke, "A growing neural gas network learns topologies", Advances in Neural Information Processing Systems 7, MIT Press, Cambridge, MA, 1995.
4. Ch. Xu, "Deformable models with applications to human cerebral cortex reconstruction from magnetic resonance images", Ph.D. Thesis, Johns Hopkins University, 1999, pp. 14-63.