A Topology-Preserving Growing Neural Network for Real-Time Representation of Objects and Their Motion

Francisco Flórez-Revuelta, Juan Manuel García-Chamizo, José García-Rodríguez, Andrés Fuster-Guilló and Jorge Azorín-López[1]

Abstract. Self-organizing neural networks endeavour to preserve the topology of an input space by means of competitive learning. This capability is used for obtaining reduced representation of objects that can be used in classification or recognition tasks. The model is extended to be also employed in tracking and motion analysis. Moreover, these applications usually have real-time constraints imposed on them. We propose the Growing Neural Gas (GNG) model due to its iterative and incremental learning process that allows. According to the available time to perform the representation, parameters of the GNG are chosen in order to finish the adaptation before the deadline. However, the quality of the representation can be affected by this parameterization, reason why we have studied topology preservation of several variants of the GNG. As self-organizing neural networks allow the learning of new patterns without beginning again the adaptation process, tracking and motion analysis is obtained by studying the dynamics of the network throughout time. We have also designed a parallel hardware architecture for our representation method, evaluating its temporal costs and processing rates by a simulation over a FPGA.

1 INTRODUCTION

Many methods to represent the shape of objects [1, 2, 3] and their motion [4, 5, 6] have been developed. However, many of the developed representations are approached from an algorithmic point of view, ignoring the aspects of implantation in a real environment where computer vision is influenced by important temporal constraints. Taking the idea from active representations [7] we employ a self-organizing neural network in order to obtain a reduced representation of the topology of the object.

Through competitive learning, self-organizing neural models adapt the reference vectors of neurons and the network that interconnects them, thereby obtaining a network that tries to preserve the topology of a high-dimensional input space [8], that is, similar patterns are mapped onto adjacent neurons and, vice versa, neighbouring neurons activate or code similar patterns. This capability is usually employed to reduce the dimensionality of an input space, extracting its most important features in order to improve or to facilitate a later classification. We use these neural models in a different way, approximating a self-organizing network to an input space of the same dimensionality and thereby obtaining a reduced representation of the data manifold, as a skeleton of the input space. Our interest lies in the final structure of the network that interconnects the neurons and that allows the representation of objects, So, the neural network is not employed to classify input feature patterns but as a characteristic to be classified.

Martinetz and Schulten [9] have formally defined what is a Topology Representing Network and its relationship with computational geometry structures as the Voronoi Diagram and the Delaunay Triangulation. So that, several models are outside this category, for instance, the broadly used Self-Organizing Feature Maps [8]. This is the reason why the first attempts to employ a self-organizing neural network as representation of objects [10] were limited to rectangular shapes, similar to the topology of Kohonen maps.

So, we have studied [11] topology-preservation of several self-organizing models to different kind of bidimensional input spaces (objects). We conclude that neural gases, as Neural Gas [12] and Growing Neural Gas (GNG) [13] are the ones that better fulfil the conditions to be a Topology Representing Network. From both of them, we employ the growing method due to its lower computational cost. As this model is incremental and iterative is very suitable for real-time object representation [14].

Self-organizing models are also capable of continuously readapting to new input patterns, it not being necessary to recommence the learning process. This latter capacity has particular applications in motion analysis [15].

These two applications - representation of objects and their motion – are, in many cases, subject to high temporal constraints, reason why the complete adaptation of the network within the available time must be assured. This is made possible by modifying the learning parameters of the GNG so as to finish within the available time. Nevertheless, this modification can affect the quality of the adaptation, measured in terms of the topology preservation of the input space. Several variants of the model to accelerate learning are considered in this paper and applied to the representation of bidimensional objects.

2 GROWING NEURAL GAS FOR REAL-TIME OBJECT REPRESENTATION

Growing Neural Gas [13] is an incremental neural model that does not require a prior specification of network size (as happens with other methods). From a minimal network, a growth process takes place that is continued until a termination condition is fulfilled. In contrast with other methods - where learning falls basically in decaying parameters - the learning parameters are constant in time.

2.1 Original GNG learning process

The GNG learning algorithm for approximating the network to the input manifold is as follows:

1. Start with two neurons a and b at random positions w_a and w_b in $\square^d$.
2. Generate an input signal ξ according to a density function $P(\xi)$.
3. Find the nearest neuron (winner neuron) s_1 and the second nearest neuron s_2.
4. Increase the age of all the edges emanating from s_1.
5. Add the square of the distance between the input signal and the winner neuron to an error counter for s_1:

$$\Delta error(s_1) = \|w_{s_1} - \xi\|^2 \quad (1)$$

[1] Dept. of Computing Technology, University of Alicante, Spain, email: florez@dtic.ua.es

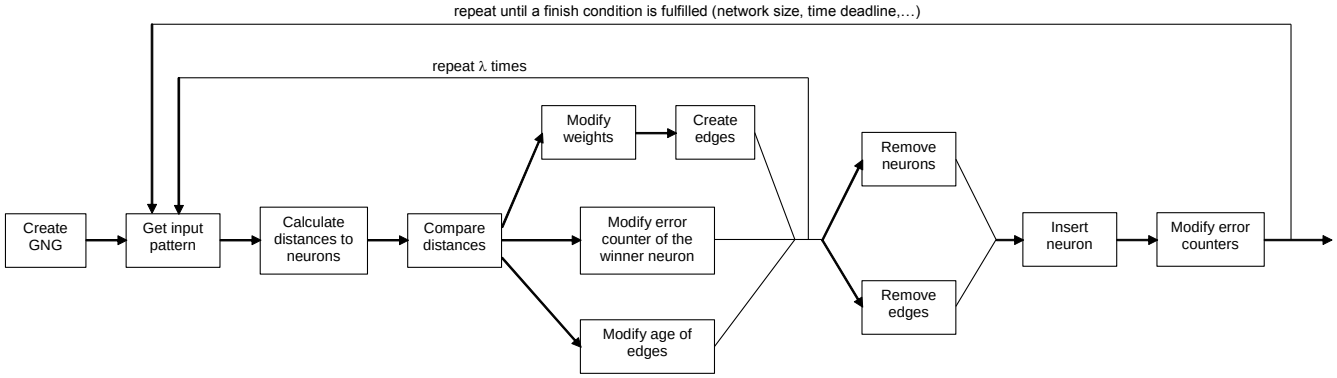

Figure 1. Graphical representation of the GNG learning algorithm.

6. Move the winner neuron s_1 and its topological neighbours (neurons connected to s_1) towards ξ according to learning steps ε_w and ε_n, respectively:

$$\Delta w_{s_1} = \varepsilon_w (\xi - w_{s_1}) \quad (2)$$

$$\Delta w_{s_n} = \varepsilon_n (\xi - w_{s_n}) \quad (3)$$

7. If s_1 and s_2 are already connected by an edge, reset the age of this edge. If no connection exists, create one.
8. Remove any edges older than a_{max}. If this results in isolated neurons (those without emanating edges), remove these as well.
9. For each number λ of input signals generated, insert a new neuron as follows:
   - Determine the neuron q with the maximum accumulated error.
   - Insert a new neuron r between q and its most distant neighbour f:

$$w_r = 0.5(w_q + w_f) \quad (4)$$

   - Insert new edges connecting the neuron r with neurons q and f, removing the old edge between q and f.
   - Decrease the error variables for neurons q and f, multiplying them by a constant α. Start the error variable of r with the new value for the error variable for q and f.
10. Decrease all error variables by multiplying them by a constant β.
11. If the stopping criterion is not yet achieved, return to step 2.

To sum up, the adaptation of the network to the input space takes place in step 6. The insertion of connections (step 7) between the two neurons nearest to each input pattern eventually establishes a Delaunay triangulation induced by the input space [13]. Elimination of connections (step 8) removes the edges that should -no longer form part of this triangulation. This is done by eliminating the connections between neurons that are no longer close by or that have nearer neurons. Finally, the accumulated error (step 5) allows the identification of those zones of the input space where it is necessary to increase the number of neurons to improve mapping.

As it is shown graphically in figure 1 learning process involves two loops: the internal one repeated a number λ of input patterns considered to insert a new neuron, and the external one that is repeated so many times as neurons are necessary to insert to complete the final size of the network.

2.2 Object representation with a Growing Neural Gas

Given an image $I(x,y) \in R$ we perform the transformation $\psi_T(x,y) = T(I(x,y))$ that associates to each one of the pixels its probability of belonging to the object, according to a property T. For instance, in figure 2, this transformation is a threshold function.

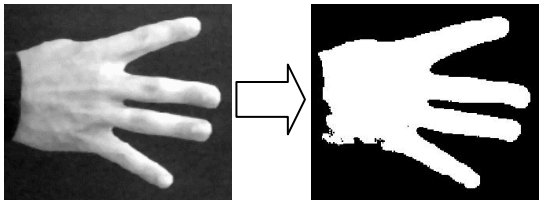

Figure 2. Silhouette extraction.

If we consider $\xi = (x,y)$ and $P(\xi) = \psi_T(\xi)$, we can apply the learning algorithm of the GNG to the image I, so that the network adapts its topology to the object. This adaptive process is iterative, so the GNG represents well the object during all the learning, giving the opportunity to stop the process if necessary, obtaining a good answer (figure 3).

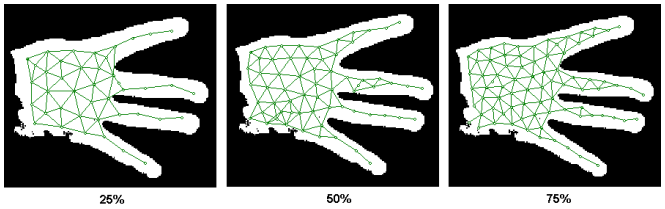

Figure 3. Adaptation process of the GNG.

As a result of the GNG learning we obtain a graph, the Topology-Preserving Graph $TPG = \langle N, C \rangle$, with a vertex (neurons) set N and an edge set C that connect them (figure 4). This TPG establishes a Delaunay triangulation induced by the object.

The model is also able to characterize the diverse parts of an object, or several present objects in the scene that have the same values for the visual property T, without having to initialize different data structures for each one of the objects (figure 5). This is due to the capacity of division of the GNG when removing neurons.

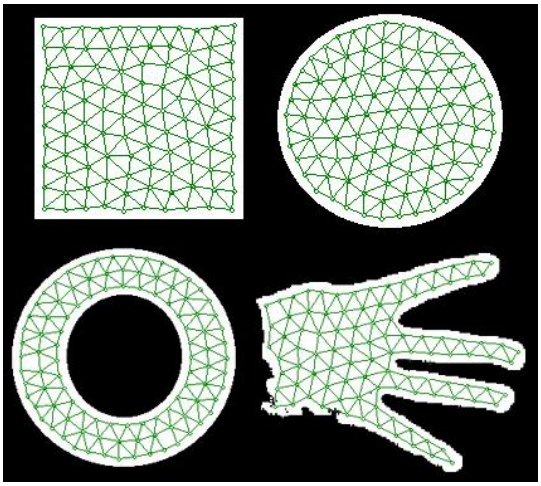

Figure 4. Representation of bidimensional objects with GNGs.

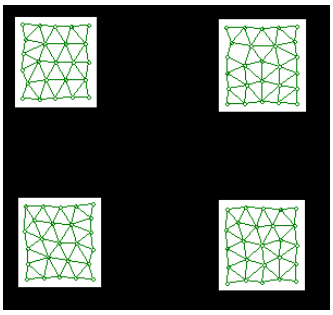

Figure 5. Representation of different fragments of an object.

The Topology-Preserving Graph that establishes a reduced representation of the object can be directly used to perform tasks of classification or recognition, or it can be employed to obtain different measures (diameter, perimeter, curvatures, signatures, chain codes, moments, subgraphs as the Gabriel graph, the minimum spanning tree,…) [14] that allow a later processing with a lower computational cost (figure 6).

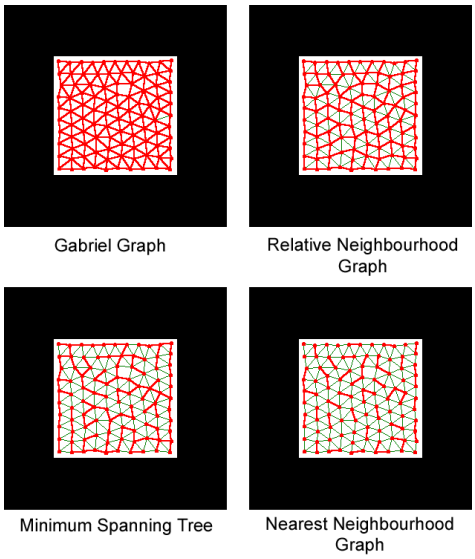

Figure 6. Subgraphs obtained from the TPG

2.3 Parameterized Growing Neural Gas

We propose the modification of the general algorithm of learning of the Growing Neural Gas With the objective of satisfying temporal constraints in the adaptation of the neural network to an input space, so that its process of adaptation will be determined by the number λ of input patterns and by the number of neurons that are inserted in each iteration. In step 9 of section 2.1 only a new neuron is inserted. Our proposal repeats this step several times. There have been already some works [16] in which more than a neuron is inserted by iteration in order to accelerate the learning of the network.

So, we define the parameterized GNG model ($GNG_{\lambda,k}$) as a variant of the original GNG network, where λ is number of input patterns considered by iteration, that is, the number of times that the internal loop is repeated; and *k* the number of times that step 9 is repeated by iteration, that implies that the network is completed performing more or less times the external loop.

On the other hand, these alternatives cause that topology preservation of the input space, that is the quality of the representation, is affected (figure 7).

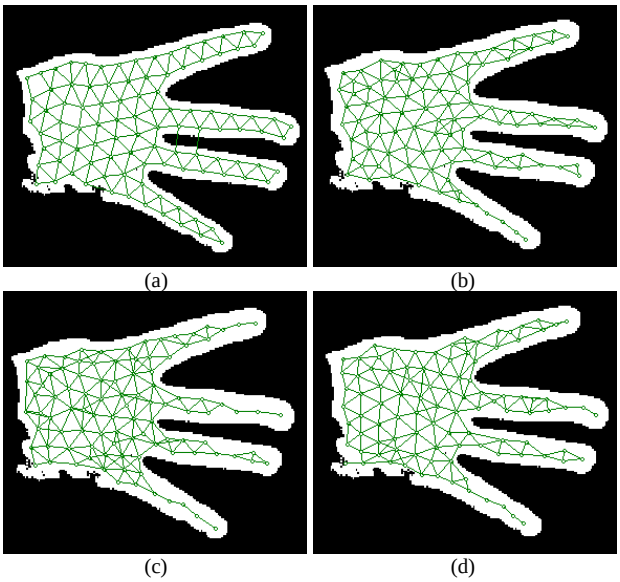

Figure 7. Final adaptations according to the number of neurons inserted per iteration: (a) 1 (original algorithm), (b) 2, (c) 5 and (d) 9

Besides, the condition to finish the learning of the network could be a deadline. So, it must be assured that the network reaches its complete size. Therefore, parameters λ y *k* must be chosen appropriately to finish the adaptation process in the available time. If such time is very low, it could happen that the network does not complete its learning and therefore incorrect/incomplete representations were achieved. Consequently, there is a loss in topology preservation due to a wrong situation of some neurons or to the existence of edges between neurons that should not be joined or the absence of others that should be created (figure 8).

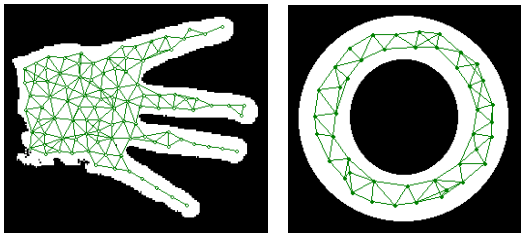

Figure 8. Incomplete adaptations due to early termination of the learning process.

3 QUALITY OF THE REPRESENTATION UNDER TIME CONSTRAINTS

Computer vision tasks must in many cases be processed with high temporal constraints determined by sampling rate. That is we have used the parameterized version of the GNG to extract a reduced representation of objects, so once the available time for the learning is known, the suitable values for parameters λ and k are chosen so that the networks finishes its adaptation. An acceleration of the learning process produces, of course, a loss in the quality of the representation, estimated with any measure of topology preservation.

3.1 Measures of topology preservation

The quality of the adaptation of self-organizing neural networks is mainly measured in two aspects: resolution and preservation of the topology of the input space.

The resolution measure usually employed is the quantization error [8], expressed as:

$$\mathcal{E} = \sum_{\forall \xi \in \mathbf{R}^d} \left\| w_{s_\xi} - \xi \right\| p(\xi) \qquad (5)$$

where s_ξ is the nearest neuron to each input pattern ξ.

The first measure developed to evaluate topology preservation was the topographic product [17], which compares the neighbourhood relationship between each pair of neurons in the network with respect to both their position on the map and their reference vectors:

$$\mathcal{P} = \frac{1}{\mathcal{N}(\mathcal{N}-1)} \sum_{j=1}^{\mathcal{N}} \sum_{k=1}^{\mathcal{N}-1} \log \left(\left(\prod_{l=1}^{k} \frac{d^{V}\left(w_j, w_{n_l^A(j)}\right)}{d^{V}\left(w_j, w_{n_l^V(j)}\right)} \cdot \frac{d^{A}\left(j, n_l^A(j)\right)}{d^{A}\left(j, n_l^V(j)\right)} \right)^{1/2k} \right) \qquad (6)$$

where j is a neuron, w_j is its reference vector, n_l^V is the l-th closest neighbour to j in the input manifold V according to a distance d^V, and n_l^A is the l-th nearest neuron to j in the network A according to a distance d^A. In order to use this measure for non-linear input spaces the geodesic distance is employed as d^V, getting what we have called geodesic topographic product [18].

Another measure is the topographic function [19], which compares the resulting neural network with the Delaunay triangulation induced by the input space, measuring the number of neurons with adjacent receptive fields but which are not connected and vice versa.

It would be desirable for all measures to take into account both aspects of resolution and topology preservation. This is not the case for the above measures, however, even though both the topographic product and the topographic function assume resolution as implicit in the competitive learning of self-organizing models.

Kaski and Lagus [20] proposed a goodness measure C that combined both aspects. First obtained is the closest reference vector for each input pattern, and then the path from that neuron to the second-closest reference vector in the map. The result is the sum of these distances.

Deviations from zero in these three measures indicate a loss in topology preservation, with the sign indicating, in the case of the product and the topographic function, whether the dimensionality of the network is larger or smaller than that of the input space to be represented.

3.2 Study of the topology preservation of the parameterized GNG

Since the calculation of these measures, particularly the most one used topographic product, has a high computational cost, it is not possible to make its calculation fast enough to process images at video rate. Nevertheless, which can be made is a study of topology preservation based on the parameters of the network and the morphology of the object to be represented. In this way, it will not be possible to assure a quality in the representation but an estimation can be achieved.

Tables 1 and 2 show the needed time to complete the adaptation of different GNGs as well as the quality of the representation to one of the objects presented in figure 6 (*hand*), employing as learning parameters: *final size=100 neurons*, $\varepsilon_1 = 0.1$, $\varepsilon_2 = 0.01$, $\alpha = 0.5$, $\beta = 0.0005$, $a_{max} = 250$, λ={1000, 2500, 5000, 10000} y k={1, 2, 5, 7, 9}. We have performed the adaptation of five net-

Table 1. Learning time and topology preservation of different GNGs ordered by learning time

Variant	Time (seconds)	Quantization Error	Geodesic Topographic Product	Topographic Function	C Measure
$GNG_{5000,1}$	2.17	57.76	0.0040	-0.10	0.198
$GNG_{5000,5}$	0.48	62.24	0.0051	-0.40	0.149
$GNG_{10000,1}$	4.25	56.80	0.0056	-0.08	0.141
$GNG_{2500,1}$	1.12	61.06	0.0067	-0.16	0.140
$GNG_{10000,2}$	2.21	59.80	0.0068	-0.10	0.151
$GNG_{10000,5}$	0.90	58.91	0.0070	-0.20	0.144
$GNG_{5000,2}$	1.13	58.98	0.0072	-0.10	0.142
$GNG_{10000,9}$	0.50	60.10	0.0073	-0.22	0.140
$GNG_{1000,1}$	0.48	61.91	0.0074	-0.20	0.140
$GNG_{2500,2}$	0.54	61.40	0.0075	-0.22	0.143
$GNG_{10000,7}$	0.65	61.83	0.0075	-0.30	0.146
$GNG_{2500,5}$	0.22	61.01	0.0093	-0.44	0.139
$GNG_{5000,7}$	0.32	61.32	0.0094	-0.42	0.148
$GNG_{5000,9}$	0.27	65.69	0.0098	-0.44	0.159
$GNG_{1000,2}$	0.24	61.32	0.0099	-0.42	0.141
$GNG_{2500,7}$	0.16	64.58	0.0111	-0.44	0.166
$GNG_{2500,9}$	0.14	63.87	0.0117	-0.50	0.158
$GNG_{1000,7}$	0.06	63.92	0.0118	-0.28	0.159
$GNG_{1000,9}$	0.06	66.60	0.0132	-0.40	0.179
$GNG_{1000,5}$	0.10	64.08	0.0138	-0.54	0.158

Table 2. Learning time and topology preservation of different GNGs ordered by geodesic topographic product

Variant	Time (seconds)	Quantization Error	Geodesic Topographic Product	Topographic Function	C Measure
$GNG_{1000,9}$	0.06	66.60	0.0132	-0.40	0.179
$GNG_{1000,7}$	0.06	63.92	0.0118	-0.28	0.159
$GNG_{1000,5}$	0.10	64.08	0.0138	-0.54	0.158
$GNG_{2500,9}$	0.14	63.87	0.0117	-0.50	0.158
$GNG_{2500,7}$	0.16	64.58	0.0111	-0.44	0.166
$GNG_{2500,5}$	0.22	61.01	0.0093	-0.44	0.139
$GNG_{1000,2}$	0.24	61.32	0.0099	-0.42	0.141
$GNG_{5000,9}$	0.27	65.69	0.0098	-0.44	0.159
$GNG_{5000,7}$	0.32	61.32	0.0094	-0.42	0.148
$GNG_{1000,1}$	0.48	61.91	0.0074	-0.20	0.140
$GNG_{5000,5}$	0.48	62.24	0.0051	-0.40	0.149
$GNG_{10000,9}$	0.50	60.10	0.0073	-0.22	0.140
$GNG_{2500,2}$	0.54	61.40	0.0075	-0.22	0.143
$GNG_{10000,7}$	0.65	61.83	0.0075	-0.30	0.146
$GNG_{10000,5}$	0.90	58.91	0.0070	-0.20	0.144
$GNG_{2500,1}$	1.12	61.06	0.0067	-0.16	0.140
$GNG_{5000,2}$	1.13	58.98	0.0072	-0.10	0.142
$GNG_{5000,1}$	2.17	57.76	0.0040	-0.10	0.198
$GNG_{10000,2}$	2.21	59.80	0.0068	-0.10	0.151
$GNG_{10000,1}$	4.25	56.80	0.0056	-0.08	0.141

works for each variant showing the average results for each measure.

Then, from these results, the system can choose the variant with better topology preservation that finishes the learning process into the available time.

As it was expected the fastest variants obtain poorer representations. There are some cases where this is not accomplished, but it is due to the random behaviour of the learning process. However, differences are very small and the representation is not significantly affected.

Moreover, GNG method, as other neural gases do, converges rapidly to the input space. So, if the system needs to interrupt the adaptation process it is very possible that topology were preserved by the network, because with a small number of neurons inserted the TPG gets a good enough reduced representation of the object. Figure 9 shows the topology preservation measured in terms of the geodesic topographic product of the variant $GNG_{10000,1}$ to the *hand* object. In the initial adaptation stages the network attempt to make a rapid representation of the input space and for this reason topology preservation fluctuates considerably. However, when a few neurons are inserted it is stabilized. So, we could consider that 100 neurons are not needed to represent the object, but a size around 50 is good enough.

4 REAL-TIME REPRESENTATION OF OBJECT MOTION

Due to the dynamic behaviour of self-organizing neural networks that allow learning new input patterns without having to restart the learning, we are going to perform object tracking and analysis of its motion by means of the temporal extension of the proposed method. This way, we model object motion from the changes that

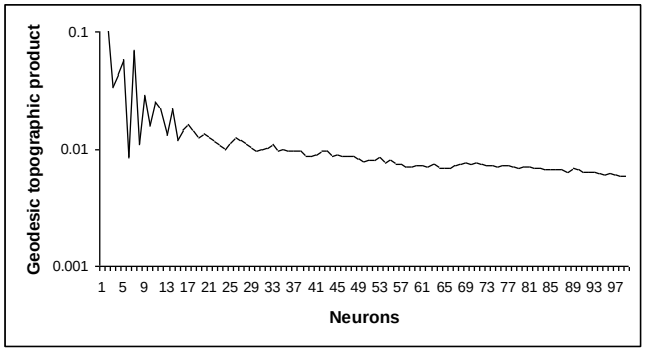

Figure 9. Topology preservation during the adaptation process.

take place in the network throughout time.

4.1 Object tracking

In order to perform object tracking, it is necessary to obtain the reduced representation of each one of the instances, understood as position and shapes, of an object for an image sequence. Nevertheless, one of the most important characteristic of our method is that if the sampling frequency is high enough, it is not required to make the complete learning of the network for each one of the images, since the TPG obtained for a given image will be very similar to the one to be got for the next one. So, with a slight readjustment of the network the new representation is achieved. This makes the tracking process very fast.

Therefore, tracking of an object in each image following the scheme presented in section 2 is composed of three steps:

- Calculation of the transformation function ψ_T to segment the object from the background.
- Prediction of the new position of the TPG
- Readaptation of the TPG

Since the conditions of the scene can vary with time, it is necessary to calculate the transformation function at every moment in order to be able to make a correct segmentation of the object.

Not only that transformation function is calculated but, since points of the object for previous frames are known because we have built the correspondent TPG, the system can update the values of property *T* that determine what is the object at the moment of segmentation. So,

$$\psi_T(x,y,t) = T\left(I(x,y,t), TPG^{t-1}\right) \quad (7)$$

Next, we get the topology-preserving graphs for each frame from that transformation functions. For the initial moment t_0 the representation is obtained following the process indicated in sections 2 and 3, making a complete adaptation of a GNG. However, for the following instances of the object the previous network is employed. So, the new representation of the object is obtained by performing the iteration of the internal loop of the learning algorithm of the GNG, without inserting nor removing neurons, simply by relocating the neurons and creating or removing edges (figure 9). This process is very fast, which allows its use in visual tracking. This readaptive method is also able to face real-time constraints, because the number λ' of times that the internal loop is performed

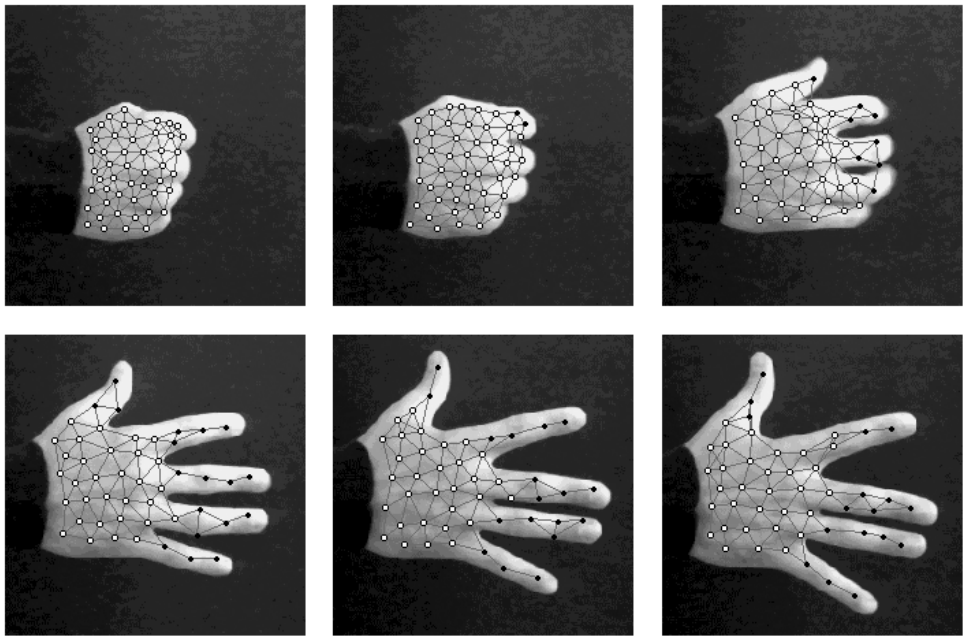

Figure 10. Dynamic adaptation of a GNG to a complete sequence. This figure presents only some frames.

can be chosen according to the available time between two successive frames.

In order to accelerate more the readaptation process, as we know speed and acceleration of the different nodes of the graph, we can predict the new position of the TPG before iterating the internal loop. This prediction approximates even more the network to the position it must get.

4.2 Motion analysis

Absolute motion (M) of an object presents two components [21]:

$$M = M_C + M_R \tag{8}$$

defined as:

- *Common motion* (M_C): motion of the whole object relative to the viewer. Basically, global translations of the object.
- *Relative motion* (M_R): motion of each one of the elements of the object relative to the other ones. Basically, morphological changes of the object.

So, the analysis of the trajectory of an object can be performed by following the centroid of the object throughout time. This centroid is calculated from the positions of the nodes of the TPG.

On the other hand, relative motion is determined by the changes in the position of each node relative to the centroid in each frame. Following the trajectories of each neuron we can analyse and recognize morphological changes in the object. We avoid one of the most important problems in tracking, that of the correspondence of relevant points between frames, because the position of the neurons is always known without extra processing.

In [15] we have used this method to hand gesture recognition. As morphological changes are mainly produced by the fingers we only consider trajectories of the nodes that are on them (black neurons in figure 10) reducing the number of trajectories to compare when recognizing the gestures (figure 11).

5 HARDWARE IMPLEMENTATION

Simulation of artificial neural networks on software platforms is suitable for testing new methods, algorithms or applications. However, their execution on personal computers or standard workstations is not fast enough to real-world problems or for big networks. Besides, due to the parallel structure of neural networks, a considerable speedup can be achieved by using parallel and custom architectures.

There are multitude of hardware implementations of self-organizing maps [8, 22, 23]. In many cases they make important simplifications of the learning algorithm in order to reduce spatial requirements if recognition or classification success rates are not affected. Other systems perform the learning off-line reducing the hardware implementation to the recognition application of the network. However, we are not interested in classification nor we are going to use the network for recognizing feature vectors. Our goal is to get a network that preserves the topology of an input space the best it could. So, we cannot make simplifications in the implemen-

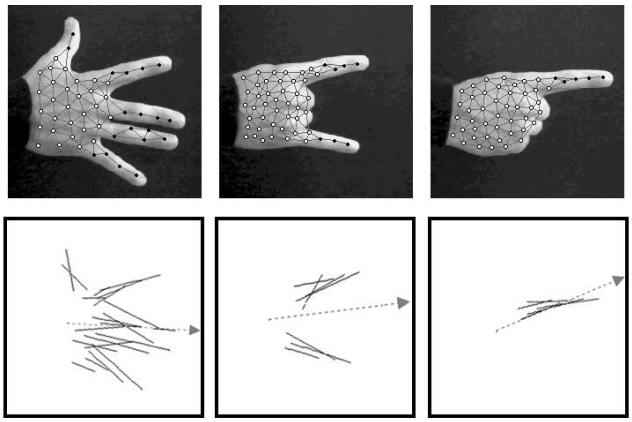

Figure 11. Analysis of the relative motion of an object. The arrow expresses the average direction of the trajectories. It is employed to do rotation-invariant gesture recognition.

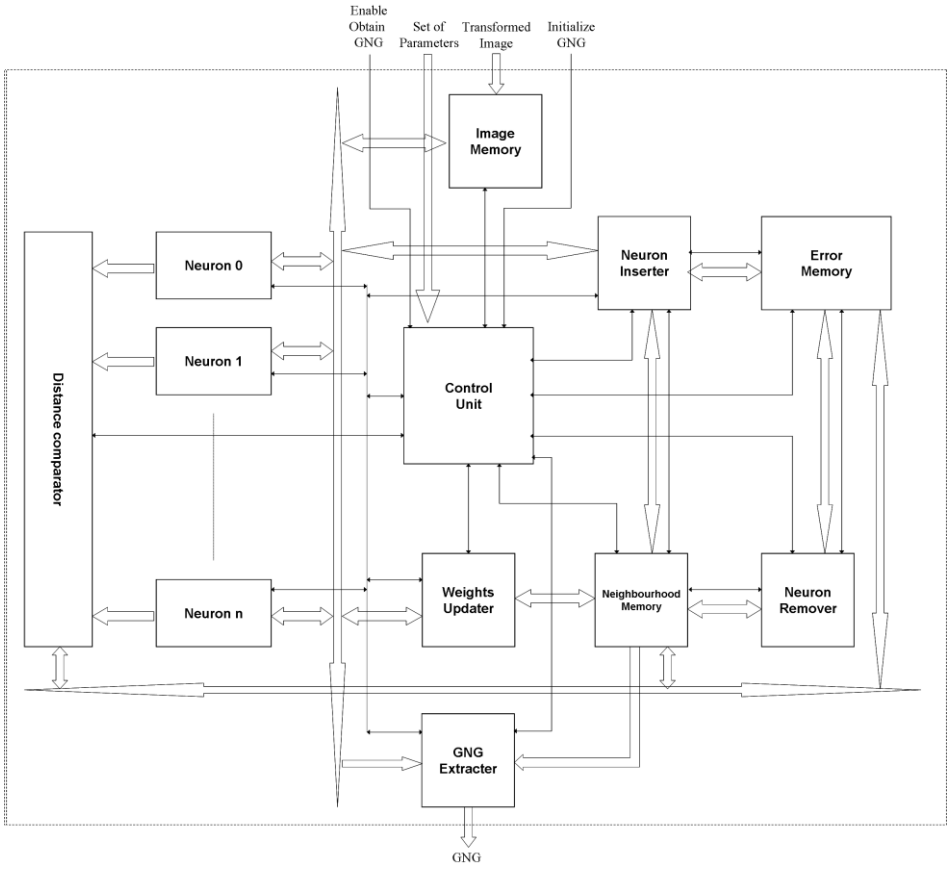

Figure 12. GNG proposed architecture.

tation. Moreover, Growing Neural Gas is a network without a restricted structure, that is neurons and edges are created and removed during the learning process. Therefore, we need to store information about connectivity of the neurons, age of the edges, number of neurons,…

5.1 Architecture

The main objective is the design of an architecture which makes use of the power of operating with all neurons in a parallel way, so as the adaptation of the network to an input space is performed in considerably lower times. The learning of the Growing Neural Gas that represents the topology of the object is carried out taking as the entry to the system the probability of each point of the image of belonging to the object, that is to say, the result of applying the transformation function ψ_T and indicating if the learning must depart from zero or it must continue from the network obtained for the previous image. Figure 12 shows a proposal for a parallel architecture following the scheme of learning of the GNG presented in figure 1.

Control Unit is the module that sends the signals to each one of the other components so that the learning is achieved. All the modules are connected to the Control Unit by an input and an output lines because it is an asynchronous architecture. This way, the modules are activated so that they begin his execution and they indicate it to the Control Unit when they finish.

One of the inputs to the Control Unit is the set of parameters of the learning process that are chosen depending on the available time to finish the adaptation. These parameters are selected according to studies that must be performed a priori to different kind of input spaces (objects) that may appear in the image, as we have done to the *hand* space in section 3.2.

Image Memory manages the transformation function to which the GNG is adapted. Its only operation is returning a random chosen *(x,y)* coordinate whose probability of belonging to the object is different from zero.

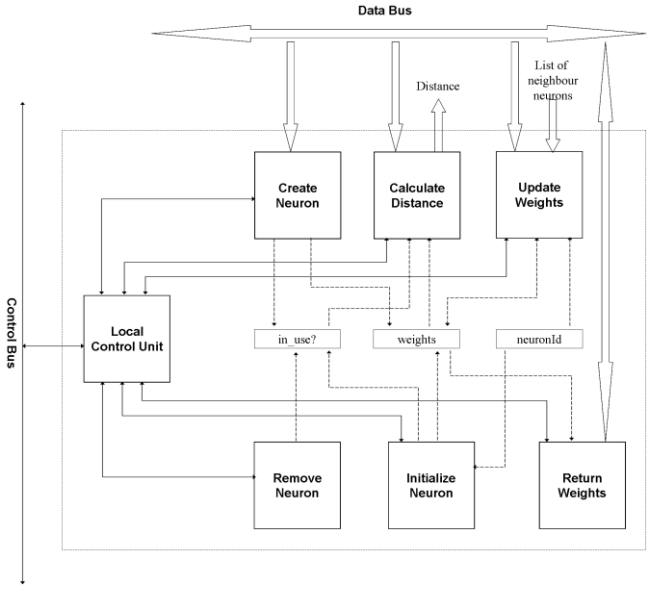

Figure 13. Neuron architecture.

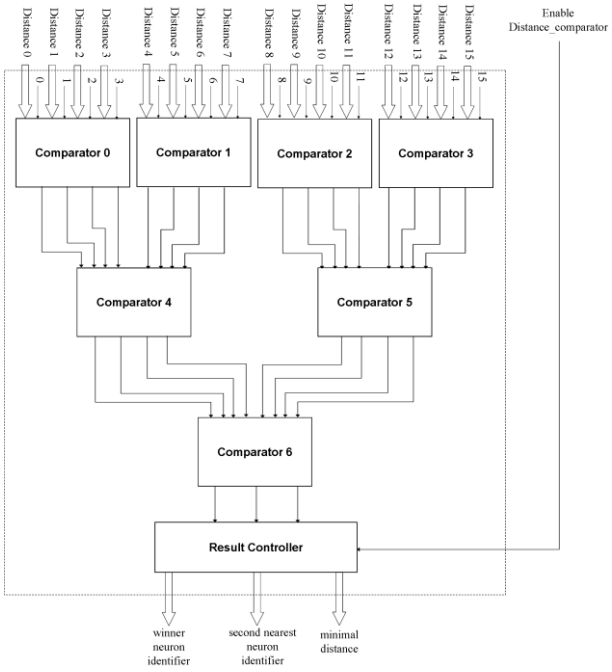

Figure 14. Distance comparator (example for a 16-neurons GNG).

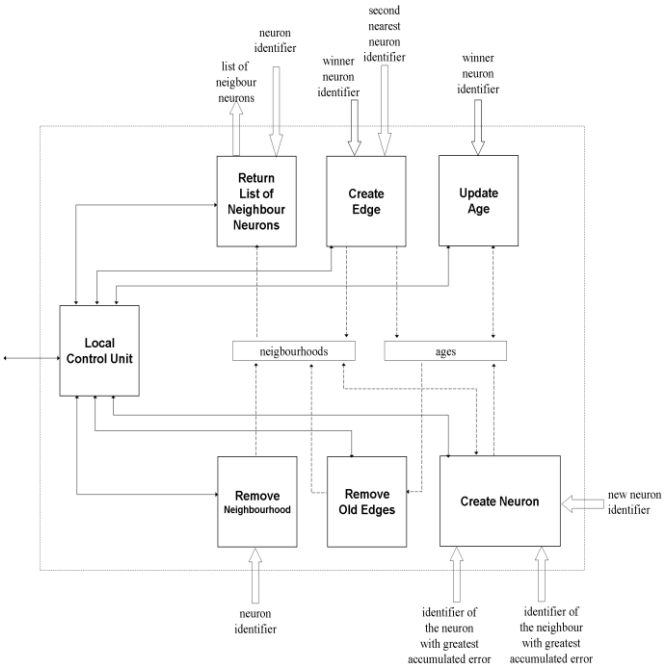

Figure 15. Neighbourhood Memory.

Each neuron has the structure showed in figure 13 and performs the following operations:

- *Initialize Neuron*: this operation takes place when the adaptation of a GNG for the first image is needed, in which the network must be initialized with two neurons. For that reason, neurons 0 and 1 initialize their weights and set that they are in use. The other neurons are set not in use.
- *Calculate Distance*: the neuron calculates the Euclidean distance from its weights to the input pattern (the coordinate).
- *Update Weights*: neuron weights are updated. This process is only carried out if it is the winner neuron or one of its neighbours.
- *Remove Neuron*: sets the neuron not in use.
- *Return Weights*: weights are placed on the data bus.
- *Create Neuron*: When a neuron is inserted, is must be set in use storing its new weights.

Distance Comparator performs the comparison of the distances obtained from all the neurons to the input pattern. As result of this process, it returns the identifiers of the winner and the second nearest neurons and the minimum distance. We have designed a tree structure of comparators in order to improve the cost of this comparison (figure 14).

Neighbourhood Memory stores information about the edges that join the neurons. It includes two number of neurons by number of neurons square matrices: one to indicate connectivity (neighbourhood) between neurons and the other one to store the age of each edge (figure 15). The operations that this module carries out are:

- *Return List of Neighbour Neurons*: It returns the row of a neuron from the neighbourhood matrix.
- *Create Edge*: Both matrices are updated to insert an edge between the winner neuron and the second nearest neuron to the input pattern.
- *Update Age*: It increments the age for every edge.
- *Remove Neighbourhood*: All the elements in the row and the column for a neuron are set to zero.
- *Remove Old Edges*: The edges that exceed a pre-established age are removed.
- *Create Neuron*: Edges are removed and inserted in a proper way when a neuron is inserted.

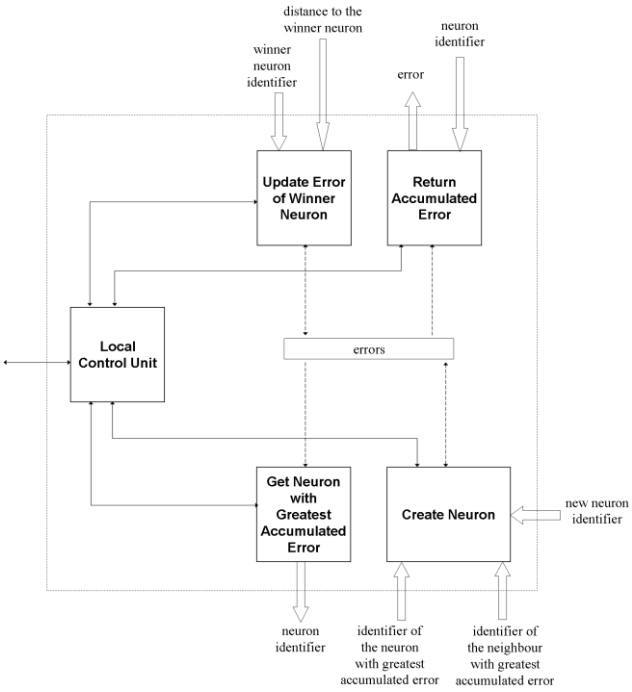

Figure 16. Error Memory.

Error Memory stores information about the accumulated error for each neuron (figure 16). In a beginning, this information was included inside each of the neurons; nevertheless, we have chosen to manage this information in an external module in order to reduce the communication with the neurons. The management of this memory is carried out by the following operations:

- *Update Error of Winner Neuron*: its value is incremented with the minimum distance to the input pattern.
- *Return Accumulated Error*: error memory is accessed and the information is sent.
- *Get Neuron with Greatest Accumulated Error*: All the stored errors are compared, returning the identifier of the neuron with greatest error.
- *Create Neuron*: when a neuron is inserted, errors are distributed between the new neurons and the two between this one is created.

5.2 Estimation of temporal cost

The proposed architecture has been defined in VHDL, simulating its processing on a standard FPGA, in order to evaluate the delays of each of the stages of the learning algorithm of the Growing Neural Gas. In particular we have employed the families of Xilinx FPGAs. The simulation and test of the design has realized with the software WebPack from Xilinx for definition, simulation and hardware implementation. We have obtained the temporal cost of this simulated execution of each one of the steps of the adaptation algorithm (tables 3 and 4).

Table 3. Temporal cost of each module of the GNG architecture.

Task	Time (ns.)
Control Unit	3.789
Create GNG	4.536
Obtain input pattern	253.526
Calculate distance	20.588
Compare distances	115.451
Modify weights	7.313
Update errors	8.586
Update edges	7.800
Remove neurons	10.524
Remove edges	9.650
Insert neuron	149.364
Get GNG	2.946

Following the path presented in figure 1 learning time for a network is:

Table 4. Learning times of the GNG architecture.

Task	Time (ns.)
Create GNG	4.536
Internal loop	420.042·λ·number_of_neurons
Inserting and removing neurons	167.47·number_of_neurons
Get GNG	2.946

With a number of neurons equal to 48, learning of a $GNG_{5000,1}$ has been performed obtaining a learning time of 96.6 ms. Since the process of initialization of the GNG is only carried out for the first image in a sequence, for the rest of the images only the internal loop must be performed. The final GNG must also be sent out to do recognition tasks. These stages have a temporal cost of 2.1 ms. per image, allowing processing rates near to 500 images per second.

The task with highest cost is "Obtain Input Pattern". This is because the generation of random input patterns must be repeated until getting a component with a probability of belonging to the object different from zero. In the tests we have carried out in this simulation 6.5 accesses to the Image Memory were necessary to get a correct coordinate. This problem can be avoided by storing the coordinates of the image that belong to the object, so this stage would have a cost of 27.839 ns. With this modification the representation of the object at the first image would be performed in 44.71 ms. and at the following ones in 0.97 ms. reaching processing rates greater than 1000 images per second.

6 CONCLUSIONS AND CURRENT WORKS

This paper introduces a new iterative, incremental and parallel method for object recognition, tracking and motion analysis based on the final configuration of a growing self-organizing neural network. We have showed how a Growing Neural Gas of the same dimensionality than that of the input space can obtain a reduced representation of an object that is in a scene. This representation, which we have called Topology Preserving Graph, can be parameterized in order to be used under real-time constraints. Due to its dynamic behaviour, tracking is performed by continuous readaptations of the network for a sequence of images. Motion analysis is carried out by studying the dynamics of the network that can be understood as the trajectories that each node of the graph follows.

In order to improve processing time we have designed the hardware implementation of the whole process. Nowadays, we are developing such design on a Field Programmable Logic Array (FPGA).

Related research is presently underway in which we are studying the degree of topology preservation for other self-organizing models (Neural Gas, GWR [24]). Our intention is to identify the characteristics of these networks that will allow a suitable and fast representation of an input space, with the objective of developing new hybrid self-organizing networks that optimize topology preservation and learning time.

By now, our approach is only valid to translations or morphological changes on the silhouette of an object. If shape does not change we can not different configurations of an object. This is the reason why we are going to introduce extra information in every neuron, such as statistics of the surroundings of the node. We expect to get the needed information to detect and analyze these kind of motions.

We are also looking for new applications of our method as the extension to higher dimensions, for instance to the representation of tridimensional objects.

ACKNOWLEDGEMENTS

This work has been partially funded by the Spanish Ministry of Education and Science under project DPI2002-04434-C04-01.

REFERENCES

[1] S. Loncaric, A Survey of Shape Analysis Techniques, *Pattern Recognition*, **31** (8), 983-1001, (1998).

[2] M. Sonka, V. Hlavac and R. Boyle, Image Processing, Analysis, and Machine Vision, Thomson-Engineering, 2nd edition (1998).
[3] R.C. Gonzalez and R.E. Woods, Digital Image Processing, Prentice Hall, 2nd edition (2002).
[4] C. Cédras and M. Shah, Motion-based recognition: a survey, *Image and Vision Computing*, **13** (2), 129-155, (1995).
[5] J.K. Aggarwal, Q. Cai, W. Liao and B. Sabata, Articulated and Elastic Non-rigid Motion: A Review, *Computer Vision and Image Understanding*, **70**, 142-156, (1998).
[6] A. Sanfeliu and J.J. Villanueva, An approach of visual motion analysis, *Pattern Recognition Letters*, **26**, 355-368, (2005).
[7] A.K. Jain, Y. Zhong and M.-P. Dubuisson-Jolly, Deformable template models: A review, *Signal Processing*, **71** (2), 109-129, (1998).
[8] T. Kohonen, Self-Organizing Maps, Springer-Verlag, Berlin Heidelberg, 3rd edition (2000).
[9] T. Martinetz and K. Schulten, Topology Representing Networks, *Neural Networks*, 7 (3), 507-522, (1994).
[10] C. Szepesvári and A. Lörincz, Approximate Geometry Representations and Sensory Fusion, *Neurocomputing*, **12** (2-3), 267-287, (1996).
[11] F. Flórez, J.M. García Chamizo, J. García Rodriguez and A. Hernández, Representing 2D objects. Comparison of several self-organizing networks, in the proceedings of the 3rd WSES Conference on Neural Networks and Applications (NNA'02), Interlaken, Switzerland, (2002).
[12] T. Martinetz and K. Schulten, A "Neural-Gas" Network Learns Topologies, in Artificial Neural Networks, T. Kohonen, K. Mäkisara, O. Simula and J. Kangas (editors), **1**, 397-402 (1991).
[13] B. Fritzke, A Growing Neural Gas Learns Topologies, in Advances in Neural Information Processing Systems 7, G. Tesauro, D.S. Touretzky and T.K. Leen (editors), The MIT Press, Cambridge, Massachussets, 625-632, (1995).
[14] F. Flórez, J.M. García Chamizo, J. García Rodriguez and A. Hernández, Representation of 2D Objects with a Topology Preserving Network, in Proceedings of the 2nd International Workshop on Pattern Recognition in Information Systems (PRIS'02), Alicante. ICEIS Press, 267-276 (2002).
[15] F. Flórez, J.M. García Chamizo, J. García Rodriguez and A. Hernández, Hand Gesture Recognition Following the Dynamics of a Topology-Preserving Network, in Proceedings of the 5th IEEE International Conference on Automatic Face and Gesture Recognition, Washington, D.C., IEEE, Inc., 318-323 (2002).
[16] G. Cheng and A. Zell, Double Growing Neural Gas for Disease Diagnosis, in Proceedings of Artificial Neural Networks in Medicine and Biology Conference (ANNIMAB-1), Goteborg, **5**, Springer, 309-314 (2000).
[17] H.-U. Bauer and K.R. Pawelzik, Quantifying the Neigborhood Preservation of Self-Organizing Feature Maps, *IEEE Transactions on Neural Networks*, **3** (4), 570-578, (1992).
[18] F. Flórez, J.M. García Chamizo, J. García Rodriguez and A. Hernández, Geodesic Topographic Product: An Improvement to Measure Topology Preservation of Self-Organizing Neural Networks, *Lecture Notes in Computer Science*, **3315**, 841-850, (2004).
[19] T. Villmann, R. Der, M. Herrmann and T.M. Martinetz, Topology Preservation in Self-Organizing Feature Maps. Exact Definition and Measurement, *IEEE Transactions on Neural Networks*, **8** (2), 256-266, (1997).
[20] S. Kaski and K. Lagus, Comparing Self-Organizing Maps, *Lecture Notes in Computer* Science, 1112, 809-814, (1996).
[21] M. Allmen and C.R. Dyer, Computing Spatiotemporal Relations for Dynamic Perceptual Organization, *Computer Vision, Graphics, and Image Processing: Image Understanding*, **58**, 338-350, (1993).
[22] T.D. Hämäläinen, Parallel implementation of self-organizing maps, in Self-Organizing neural networks: recent advances and applications, *Springer Studies In Fuzziness And Soft Computing Series*, 245-278, (2001)
[23] M. Porrmann, M. Franzmeier, H. Kalte, U. Witkowski and U. Rückert, A Reconfigurable SOM Hardware Accelerator, in Proceedings of the European Symposium on Artificial Neural Networks, Bruges, Belgium, 337-342 (2002).
[24] S. Marsland, J. Shapiro and U Nehmzow, A self-organizing networks that grows when required, *Neural Networks*, **15**, 1041-1058, (2002).