Proceedings of the 2009 IEEE International Conference on Systems, Man, and Cybernetics
San Antonio, TX, USA - October 2009

The Use of TurSOM for Color Image Segmentation

Derek Beaton
James J Kaput Center for Research and Innovation in Mathematics Education
University of Massachusetts Dartmouth
Fairhaven, MA 02719

Iren Valova, Dan MacLean
Computer and Information Science Department
University of Massachusetts Dartmouth
North Dartmouth, MA 02747

Abstract—**This work presents an application of TurSOM for high-dimensional segmentation and cluster identification. TurSOM is a variation algorithm of the SOM algorithm, which introduces a new mechanism of self-organization: connection reorganization. We have theoretically presented TurSOM in very recent previous work, however, the applicability of the novel architecture is expanding as we explore it numerous advantages and possibilities. The intent of these experiments, and TurSOM itself, is to be able to segment and identify various distinct objects in color data (red, green, blue).**

Keywords—**Self-organizing Maps, Turing Unorganized Machines, unsupervised learning, clustering, segmentation**

I. INTRODUCTION

TurSOM [1, 2, 3, 4] is a self-organizing map algorithm that introduces a very new, and unique aspect of self-organization – connection reorganization. TurSOM is inspired by the initial self-organizing map algorithm proposed by Kohonen [5], and the model of the human cortex proposed by Turing, called unorganized machines [6].

The original rules of neuron learning from Kohonen's algorithm still apply to the neurons in TurSOM, however, TurSOM introduces some new neuron features, as well as several new features to establish self-organization in connections, which facilitates a multi-network behavior. One of the most significant features of connection reorganization is that connections between neurons may be removed entirely, allowing multiple networks to compete for input within a single input space. This behavior has two major advantages: 1) it happens during execution, and 2) it facilitates separate networks discovering and maping distinct and unique patterns in input space.

TurSOM's multiple network behavior serves as an in-execution method of clustering and pattern identification. TurSOM has been used previously for simple two dimensional pattern identification [1, 2], gray scale image segmentation [1, 3], and preliminarily for higher dimensional work [4], which we are building upon in this work.

II. TURSOM

A. Background

TurSOM exhibits two forms of competitive learning by simultaneously reorganizing both neurons and connections. The first level of competitive learning exists within the neurons, as Kohonen's algorithm does, by neurons competing for input (best matching unit - BMU). However, TurSOM has a mechanism, known as the gap junction (GJ), which facilitates connection reorganization. Since a connection between neurons can be removed altogether, this facilitates the second level of competitive learning. When a connection between neurons is removed, only the neurons within that network are affected, leading to the selection of the best matching network (BMN). The compartmentalized details of TurSOM are broken down, and elaborated on in the following sections.

B. Reorganization

Traditionally, the connections between neurons in SOM algorithms provide path and neighbor information for the winning neuron. These connections, for the most part, exist to bind neurons together most commonly in a one-dimensional chain (neurons have at most two neighbors) or a two-dimensional lattice structure (grid or hexagonal).

TurSOM's primary component, and most significant contribution, is that it adds functionality to the connections. Connections now have a mechanism called the gap junction (GJ), which observes: 1) the length of the connection it resides on; and 2) connections that are nearby. Both attributes are measured by neuron positions. As the GJ observes the length of the connection it resides on (figure of gap junction), if the length becomes too long, with respect to the current network it is in, the connection is severed, effectively splitting the current network in two.

Additionally, connections can exchange places, if the neurons they are connecting would be better off with other neighbors. This exchange happens between two connections and three to four neurons (depending on the number of neurons in the region).

These features were introduced to account for and correct non-linearity in connections, which our previous work [7, 8, 9 10] indicates that "tangled" networks can cause convergence issues, as well as misrepresent the underlying data.

C. Neuron Responsibility

Neurons in TurSOM exist in nearly the same capacity as they do in a traditional SOM algorithm. However, there is one major feature that has been added to the neurons of TurSOM that requires explanation. Neurons that are effectively "freed" from a connection – that is, the connection it had with another neuron that was removed due to (effectively) incompatibility in attributes now have a Neuron Responsibility Region (NRR). Essentially, all neurons that do not have the maximum number of neighbors possible enter a state where they are actively seeking new neighbors to complete their connections. In order to seek for new neighbors, a neuron needs to look in a radius beyond where it exists, and must find another neuron that is also actively looking for another connection.

The neuron responsibility radius is determined by the number of available inputs and the number of total neurons that exist in the input space. This value is dependent on the dimensionality of attributes, and subsequently, if that dimensionality is even or odd numbered. If *n* is even, where *n* is the number of attributes:

$$r_e = \left[\rho / e \right]^{1/\delta} \tag{1}$$

where *e* is:

$$\mathrm{e} = \left(\frac{\delta}{2}! \right)^{-1} \times \pi^{\frac{\delta}{2}} \tag{2}$$

If the value of *n* is odd:

$$r_o = \left[\rho / o \right]^{1/\delta} \tag{3}$$

where o is:

$$\mathrm{o} = \left(\frac{2^{\delta} \times \frac{\delta - 1}{2}!}{\delta !} \right) \times \pi^{\frac{\delta - 1}{2}} \tag{4}$$

Where δ represents the number of dimensions and ρ is calculated by dividing the number of inputs by the number of available neurons. ρ is the number of theoretical inputs a neuron is responsible for.

D. Connection Responsibility

Connection responsibility is governed by the GJ mechanism. The term gap junction is borrowed from biology, where it represents a mechanism that is found in neural cells and is responsible for the bi-directional communication of electrical signals between neurons. The gap junction of TurSOM, however, is not biologically accurate, as TurSOM does not model electrical or chemical relationships like ART3 [11]. The GJ of TurSOM is biologically *inspired*, as it plays the role of communicating information between neurons. The GJ decides if, and when a connection will either remove itself from the network, or exchange positions with another connection.

This is the primary and most significant behavior of TurSOM. As Kohonen posited, given enough iterations a one-dimensional SOM will converge with linearity (no tangles), effectively in a Peano-like curve. However, two-dimensional networks have no such guarantee. In previous work [8, 10] we have shown that initialization techniques that are already Peano-like (Hilbert curves), converge faster than random initialization, and have no tangling in the network. TurSOM circumvents the requirement for a non-tangling initialization method because the connections reorganize in execution.

1) Connection Learning Rate

The gap junction of TurSOM is responsible for computing what is known as the connection learning rate (CLR). This is a value that is analogous to the neuron learning rate with one critical difference: as a neuron learning rate progresses it gets smaller – causing neurons to move less and less, whereas the connection learning rate increases over time, allowing more stringent and stable connection behavior.

The connection learning rate is a measure that calculates the length of the connection – determined by the neurons it connects – and then computes the "relative bigness" of that connection with respect to all other connections in that network – not all connections that exist in the input space. The relative bigness is computed by using a modified version of the standard upper outlier formula, most prominently used for boxplots:

$$CLR = Q_3 + (i \times (Q_3 - Q_1)) \tag{5}$$

In the standard upper outlier formula, *i* is either 1.5 for mild outliers, or 3 for extreme outliers. In TurSOM however, it is an incrementing value. As the network progresses, the value of *i* increases, which means that in the early iterations of TurSOM, connections can be quite small and rapidly reorganize, and in late iterations a connection must become quite large in order to remove itself from the network. The connection learning rate does not influence connection reorganization, however. Connection reorganization is determined by neuron positions and is used a method to prevent network tangling.

E. Algorithmic Explanation

TurSOM's algorithmic steps being with the same as Kohonen's SOM algorithm, and this section will provide explanation only for the new steps and features that are exclusive to TurSOM. Just like Kohonen's algorithm, selecting input, finding the best matching unit via competitive learning, and moving a unit and its neighbors toward the input are the first steps. One significant detail to note is that with TurSOM, the selection of best matching unit also indicates the selection of a best matching network. The following steps are unique features for TurSOM:

1) Check connection lengths

All the connection lengths in the input space are checked to see if they are longer than the value that is currently held by the connection learning rate variable. If a connection is larger than the CLR value, that connection is removed from its network and subsequently causes two separate networks to exist.

When disconnection occurs, *i*, of CLR is incremented, effectively increasing the CLR value. Additionally, any neurons that were subject of disconnection now have their responsibility radius (NRR) activated, and search for other neurons with a free connection that are within the radius.

2) *Connection Exchanges*

In addition to connections being removed or added (when two actively seeking neurons find each other) to the network, connections are also exchanged between neurons to optimize neighbors in a network.

Connections are exchanged between neurons in addition to the removal and addition of connections, because neurons may have better neighbors. Connection exchanges also avoid network tangling. Given three neurons: A, B, and C where neurons A and B are *directly* connected to one another, and C is in either the same or a different network from A and B, connections will be exchanged if the topological attributes of A and B are *less* similar than either A is to C or B is to C.

III. Growing

The algorithmic and feature descriptions of TurSOM provided in Section II are for a one-dimensional network. One-dimensional networks have advantages and disadvantages over two-dimensional lattice structures in SOM networks. Firstly, the disadvantages are that one-dimensional networks may not provide accurate topological information, and will not in fact be representative of the underlying data. The features of TurSOM are not necessarily limited to one-dimensional networks, but are implemented only for one-dimensional networks.

To ensure that TurSOM is a robust SOM that will accurately represent underlying data, a growing mechanism has been introduced. The growing mechanism that TurSOM has is a single growth effect similar to that of growing grids [12] introduced by Fritzke. Prior to explaining the growing mechanism of TurSOM, we shall briefly review highly regarded growing SOM networks.

A. *Growing Architectures*

Growing architectures are most significantly attributed to the work of Fritzke [12,13,14]. In more recent SOM history, other algorithms including ParaSOM [15] and ESOINN [16] also have growing mechanisms. The following subsections include brief, qualitative descriptions of these algorithms and the growing mechanisms contained therein.

1) *Fritzke's Architectures*

Growing cells (GC), growing grids (GG), and growing neural gas (GNG) are all algorithms attributed to Fritzke. The fundamental behavior of all of these algorithms is that they start out with a minimal network size and continue to grow until some user-defined threshold is met. This user threshold can be number of neurons, or number of iterations that the network is executing. Each of these algorithms possesses similar features that allow the algorithms to grow or behave in different ways.

Fritzke's algorithms all contain age parameters, either for the neurons (GG, GC) or for the connections (GNG). As the age parameter increases (as a consequence of winning via competitive learning), more neurons and edges are introduced to those with high age values. High age values indicate a region of input space that is apparently dense with input, as selection in that region is quite high. The age parameters will be less over time as there are now more neurons or edges to distribute over that area.

2) *ParaSOM*

ParaSOM is another growing algorithm that also exploits the inherent parallel nature of the SOM algorithm. ParaSOM also has unique features of its neurons – the neurons have a Gaussian radius within which they are responsible for all input under that distribution. Similar to Fritzke's algorithms, ParaSOM has a growing mechanism that is based largely on age of the neurons.

3) *ESOINN*

ESOINN (enhanced self-organizing incremental neural network) [16] is a newer architecture that utilizes a growing mechanism. During execution, ESOINN (and similarly SOINN – ESOINN's predecessor) add or remove connections based on criteria of density and age of nodes and connections.

ESOINN's behavior however, is different from the other architectures in that it uses a two best-matching unit methodology. ESOINN's competitive process selects the best unit for the current input, as well as the next best unit.

ESOINN's connection building and removal relies on density information provided by the neurons in the network. Connection addition occurs when two best-matching units are either in the same subclass, or if one of the nodes is a newly grown node, introduced by growing mechanisms. In the final stages of the network, nodes are reclassified for optimal class identification, which will consequently remove connections between nodes that are determined to be from different sub-classes.

B. *Growing Mechanism in TurSOM*

TurSOM has a very different growing mechanism, and even a vastly different initialization method than other algorithms mentioned in this paper. A major difference between TurSOM and all other architectures is that TurSOM begins as a statically sized one-dimensional network. As it adapts to input, new nodes are never introduced – only connections and neurons are self-organizing.

Effectively TurSOM uses a growing mechanism that is semi-analogous to Fritzke's GG. The GG algorithm adds either entire rows, or entire columns of neurons. However, TurSOM grows only once. The growing mechanism can be instantiated by either a user-defined iteration value, or based on a value measuring one-dimensional convergence.

In TurSOM's growing mechanism, both rows and columns are added to the one-dimensional network. Growing transforms the previous one-dimensional sub-networks into grids of a square size. For example, a one-dimensional sub-network of size 10 will become a grid of neurons in a 10x10 lattice structure.

To maintain unsupervised, and unbiased approach to placing neurons in input space, after growth occurs, TurSOM simply increments attributes by a very small number, so that neurons are distinct, but very similar to their predecessor (previous row or column). Effectively, the initial row is duplicated and shifted in either all, or just one dimension by the same incrementing value.

IV. Initialization & Phases

TurSOM places neurons for initialization with one of two methods: 1) random vectors in input space or 2) Hilbert-curve initialization [8,9,10]. Hilbert-curve initialization is a method developed by the authors to place neurons in input space, without having any network tangling at initialization. In addition, we have shown that preventing network tangling at initialization leads to faster convergence, and little to no network tangling during execution.

However, tangling issues are of no concern to TurSOM during the second phase of execution: connection-adaptation phase.

A. Network Adaptation Phase

TurSOM has a user-defined parameter to indicate which phase it should begin with. The network-adaptation phase is effectively the standard SOM map algorithm and behavior. Network-adaptation (NAP), does not allow connections to reorganize, simply it allows neurons to self-organize. It is done as a preliminary step to connection-adaptation, wherein connections are allowed to reorganize.

B. Connection-Adaptation Phase

The connection-adaptation phase (CAP) of TurSOM is effectively the primary, and most important phase of TurSOM, wherein the connections become active learners and critical to the self-organizing process of all sub-networks.

CAP transition is a user-defined parameter, in TurSOM's current form, however, nothing bars CAP transition from being instantiated by a convergence metric to indicate that the (whole single original) network is no longer a viable learning tool.

CAP often occurs at network instantiation, which would completely replace NAP. However, the user may indicate that TurSOM should operate as a regular SOM for a given amount of iterations or until a certain level of convergence has been reached.

V. Experiments

The following experiments build upon prior works [1, 2, 3, 4], with the intent of taking TurSOM to scale for high-dimensionality, as well as a more user-friendly product, where the number of user definable parameters is reduced.

Previous work [1, 2] has shown that TurSOM is a unique contribution to the SOM-variant field. The most significant contributions of TurSOM have been a reduction in execution time (measured in iterations), and in-execution network partitioning. In-execution partitioning is effectively a form of clustering. Traditionally, prior work with SOMs (and many variants) required a post processing technique (such as PCA) to group neurons together, or that the number of nodes of a SOM were limited, and those nodes represent clusters in a fashion that would be similar to k-means or a non-linear PCA.

This paper is presenting an extension of previous work [3, 4] with respect to high-dimensionality. In [3], and to a lesser extent [1] (the origin of TurSOM) we present TurSOM as a method of image segmentation and clustering. Dimensionality did not exceed three dimensions (x and y locations and gray scale value of pixels). Here, we extend this to larger input space with more intricate patterns, specifically, concentric color-degrading circles, and an extension of two benchmark patterns, the double-spiral and the circle-ring pattern.

Both experiments presented here use random vector initialization for the x and y locations, whilst the red, green, and blue (RGB) values are initialized to 127, effectively rendering the neurons as gray pixels.

A. Concentric Color-Degrading Circles

This experiment uses a pattern that uses seven concentric circles in a degrading pattern, beginning with black, and followed by the colors found in a spectrum, effectively that of a rainbow: red, orange, yellow, green, blue, violet. This pattern is displayed in Fig.1.

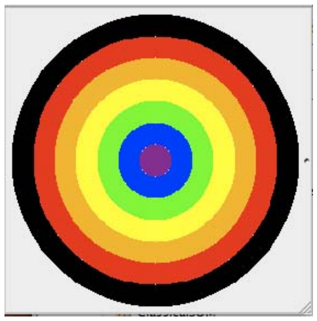

Figure 1. The original pattern of concentric color-degrading circles

The following series of figures present TurSOM, in one-dimensional form, adapting to this pattern. Initialization of TurSOM for this pattern has 200 neurons as shown in Fig.2.

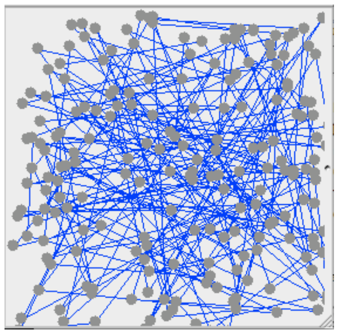

Figure 2. Initialization of TurSOM with 200 neurons to random vectors.

Fig. 3 demonstrates TurSOM during the first phase – network-adaptation. Network-adaptation includes two phases, wherein the first phase includes a high neighbor hood radius, to effectively collapse the network and reduce tangling. The second phase has a low neighborhood radius.

Fig. 4 shows TurSOM still in mode of network adaptation. Neurons are approaching vectors, but the network is still highly tangled. The behavior at this stage is no different than a normal SOM. However, if a normal SOM were too continue execution from this state, the amount of required iterations until Peano-curve like convergence are very high [3].

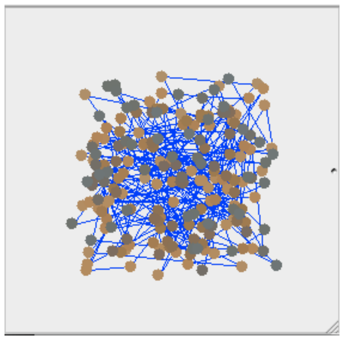

Figure 3. 100 Iterations into TurSOM execution

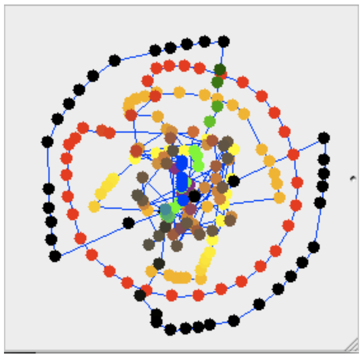

Figure 4. TurSOM after 500 iterations

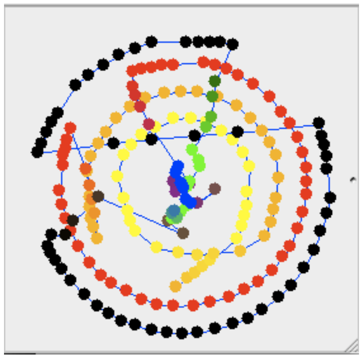

Figure 5. TurSOM in iteration 1250, still in network-adaptation.

Fig. 5 demonstrates TurSOM in iteration 1250, still in network-adaptation phase. Fig. 6, in iteration 1800 demonstrates TurSOM after connection-adaptation phase was turned on in iteration 1251. This displays the in-execution clustering behavior of TurSOM. The mechanisms of this behavior are elaborated on in [1, 2].

Fig. 6 shows TurSOM in iteration 1800, well into connection-adaptation. However, comparable to network-adaptation and traditional SOMs, TurSOM is capable of quickly identifying distinct patterns.

B. *Double-Spiral Circle-Ring pattern*

This pattern is based on two benchmark patterns, the double-spiral and the circle-ring patterns. Each are common benchmark patterns for clustering and distinguishing between overlapping data in the same input space. Below is how the pattern looks without a network present

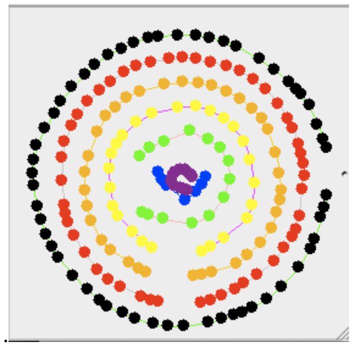

Figure 6. TurSOM in iteration 1800

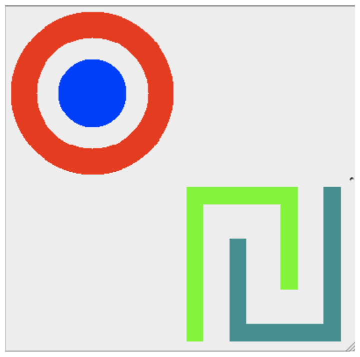

Figure 7. Circle-Ring and Double-Spiral with various colors and overlapping x and y coordinates.

Fig. 7 demonstrates the patterns as they exist in space. In this experiment, the growing mechanism of TurSOM will also be demonstrated. Fig. 8 displays TurSOM at initialization. This experiment uses 100 neurons for the network. Few neurons are needed at initialization when the growing mechanism of TurSOM is used, as the number of neurons will, at some iteration (defined by the user) will increase to better map patterns.

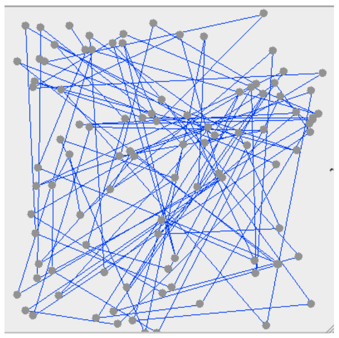

Figure 8. TurSOM with 100 neurons at initialization with random vectors.

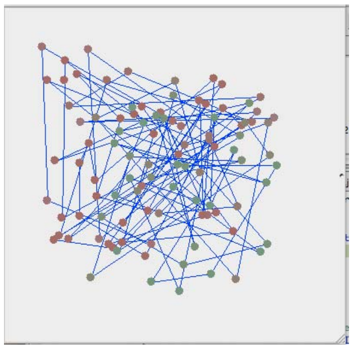

Figure 9. TurSOM at iteration 100 during network-adaptation phase.

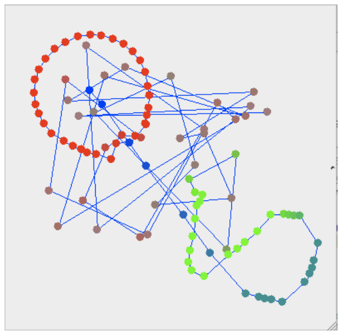

Figure 10. TurSOM at iteration 250 during network-adaptation phase.

Figs. 9 and 10 show TurSOM at iterations 100, and 250 respectively. The behaviors are similar to the prior experiment. In this experiment, the first phase of network-adaptation lasts for 200 iterations. Fig. 10 shows TurSOM in a state that is still very like a traditional SOM.

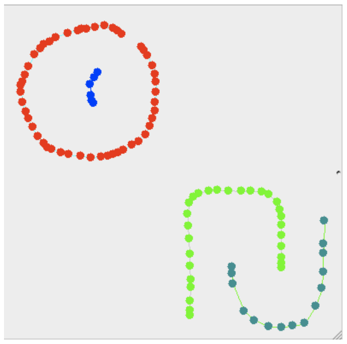

Figure 12. TurSOM at iteration 850. Connection-adaptation has been on-going for 99 iterations.

Figs. 12 and 13 demonstrate TurSOM in it's fully featured mode. As Fig. 12 shows, TurSOM discovers distinct patterns in five-dimensional space. Fig. 13 demonstrates the effects of the growing mechanism of TurSOM, approximately 250 iterations into execution.

VI. Conclusion & Discussion

In this paper we present new application of TurSOM that demonstrates the capability of high dimensional pattern segmentation. Color image segmentation is a very powerful tool, and TurSOM is a simple, yet effective method of providing it. TurSOM is not a vast variant of the traditional SOM algorithm, yet introduces certain features that make it far more powerful as a pattern recognition and identification tool.

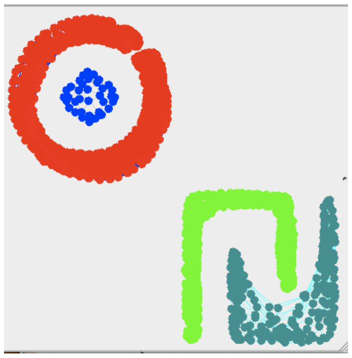

Figure 13. TurSOM at iteration 1250. The growing mechanism was turned on at iteration 1001.

References

[1] D. Beaton, "Bridging Turing Unorganized Machines and Self-organizing Maps for Cognitive Replication", UMass Dartmouth MS Thesis, 2008

[2] D. Beaton, I. Valova, D. MacLean, "TurSOM: A Turing Inspired Self-organizing Map," accepted to IJCNN, 2009, in press.

[3] D. Beaton, I. Valova, D. MacLean, "Growing Mechanisms and Cluster Identification with TurSOM," accepted to IJCNN, 2009, in press.

[4] D. Beaton, I. Valova, D. MacLean, "Color Objects Identification with TurSOM," accepted to ICCNS 2009, in-press.

[5] T. Kohonen. *Self-Organizing Maps*, Springer, second ed., 1995.

[6] D.C. Ince (ed.). *Collected Works of A.M. Turing – Mechanical Intelligence*, Vol. 3/4 Elsevier Science Publishing, 1992

[7] D.Beaton, I.Valova, D.MacLean, "CQoCO: a Measure for Comparative Quality of Coverage and Organization for Self-Organizing Maps," *Neurocomputing (in review), Elsevier.*

[8] I.Valova, D.Beaton, A.Buer, D.MacLean, "Fractal Initialization for High-Quality Mapping with Self-Organizing Maps," *Journal of Neural Computing and Applications (in review), Springer.*

[9] Iren Valova, Daniel MacLean, Derek Beaton, "Identification of Patterns via Region-Growing Parallel SOM Neural Network," icmla,pp.853-858, 2008 Seventh International Conference on Machine Learning and Applications, 2008

[10] I. Valova, D. Beaton, D. MacLean, "Role of Initialization in SOM Networks - Study of Self-Similar Curve Topologies", *Proceedings International Neural Conference on Artificial Neural Networks in Engineering (ANNIE), Vol.18, pp 681-688, 2008*

[11] G. Carpenter, S. Grossberg. "ART 3: Hierarchical Search Using Chemical Transmitters in Self-organizing Pattern Recognition Architectures," *Neural Networks,* Vol. 3, pp. 129-152, 1990.

[12] B. Fritzke. "Growing Grid – a self-organizing network with constant neighborhood range and adaptation strength," *Neural Processing Letters*, Vol 2 No. 5 9-13 1995.

[13] B. Fritzke. "A Growing Neural Gas Network Learns Topologies," *Adv. In Neural Information Processing Systems*, Vol. 7 pp. 625-632, 1994.

[14] B. Fritzke. "Kohonen Feature Maps and Growing Cell Structures – a Performance Comparison," *Advances in Neural Information Processing Systems*, Vol. 5, 1993.

[15] I. Valova, D. Szer, et al. "A parallel growing architecture for self-organizing maps with unsupervised learning," *Neurocomputing*, Vol. 68 pp. 177-195, 2005.

[16] S. Furao, T. Ogura, O. Hasegawa. "An Enhanced Self-Organizing Incremental Neural Network for Online Unsupervised Learning," *Neural Networks*, Vol. 20 No. 8, pp. 893-903, 2007.