1

An Adaptive Fuzzy Neural Network Based on Self-Organizing Map (SOM)

Jun-fei Qiao and Hong-gui Han
Beijing University of Technology
China

1. Introduction

This chapter shows a new method of fuzzy network which can change the structure by the systems. This method is based on the self-organizing mapping (SOM) (Kohonen T. 1982), but this algorithm resolves the problem of the SOM which can't change the number of the network nodes. Then, this new algorithm can change the number of fuzzy rules; it takes the experienced rules out of the necessary side for the number of the fuzzy rules. We use this new algorithm to control the dissolved oxygenic in the wastewater treatment processes. This proposed algorithm can adjust subjection function on-line, optimize control rules. The results of simulations show that the controller can take the dissolved oxygenic to achieve the presumed request, and prove the superiority of this proposed algorithm in the practical applications.

The research of the structure of the Neural Network is a hotspot currently. A neural network model with strong relations to the area of fuzzy systems is the fuzzy neural network model (T.Poggio and F.Girosi, 1990). Based on the IF-THEN rules, the fuzzy logic rules can be clustered. The functional equivalence of restricted fuzzy neural networks has been shown as Fig1:

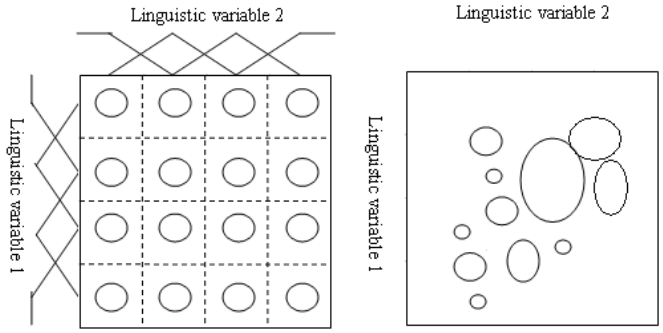


Fig. 1. Relations of the fuzzy linguistic before and after clustering algorithm

Fig (a) is a restricted fuzzy system which distributes the input into a number of fuzzy regions. Fig (b) is the relations of fuzzy linguistic after clustering. Fig (a) and Fig (b) show that the relations of fuzzy linguistic before and after clustering are not the same. In fact, the functions of the fuzzy rules are also not the same in the general control. Especially in the fuzzy neural network (Yu Zhao, Huijun Gao & Shaoshuai Mou, 2008), every node represented a rule in rules layer. But some edge rules are even not be used in the training phase. And every rule must be computing at every training phase, so the time will be lost. In some real-time control systems the conventional fuzzy neural networks can hardly meet the requirement.

In order to solve this problem how to ensure the number of network nodes, Fritzke devised one of the first growing neural networks which is Growing Cell Structure (GCS) (Fritzke, B. 1994). The GCS is based on the SOM (Kohonen T. 1982), the network predefines some rules. A new node is inserted when accumulated error is higher than the predefined parameter, and the network continues to adapt and grow until some stopping criterion is met. This can take the form of a network size. But this structure also has limitation which can only add nodes but can not reduce them.

Merkel and his partners invented Growing Hierarchical Self-Organizing Map (GH-SOM) (Dittenbach. M., Merkel. D & Rauber. A., 2000). The key idea of the GH-SOM is to use a hierarchical structure of multiple layers where each layer consists of a number of independent Self-Organizing Mapping (SOM). The network inserts complete rows or columns when the average error higher than a constant parameter and reduce a node when the average error lower than a constant parameter. The nodes can be added and reduced in this network, so this network structure can solve the former limitations. But the number of the network nodes may be too high about some simple systems. Besides, there are other growing strategies for constructing the RBF neural network (Liying Ma & K. Khorasani, 2005; R. Sentiono, 2001; S. S. Ge, F. Hong, & T. H. Lee, 2003).

Based on the GCS and GH-SOM a new Adaptive Growing Self-Organizing Fuzzy Neural Network (SFNN) will be introduced in the following sections. The number of the conventional fuzzy neural network can be changed by this method. The details of this method can be found in paper (Junfei Qiao, Honggui Han, and Yanmei Jia, 2007).

The organization of this chapter is as follows: we describe the growing self-organizing fuzzy neural network (GSFNN) in the next section. And we divide into three sections in this part to discuss this problem. In Section III, we introduce the controller based on the growing self-organizing fuzzy neural network (GSFNN). In Section IV, we use this proposed algorithm to control the dissolved oxygenic (DO) in the wastewater treatment process and compare the results with the conventional fuzzy neural algorithm. Finally, the merits of the proposed growing self-organizing fuzzy neural algorithm are shown, and the conclusion is given in Section V.

2. Growing Self-Organizing Fuzzy Neural Network (GSFNN)

This section consists of three main parts: The growing self-organizing algorithm; the fuzzy neural network and the growing self-organizing fuzzy neural network. The growing self-organizing algorithm will be described in section2.1; the fuzzy neural algorithm adaptive self-organizing fuzzy neural network will be shown in section 2.2 and the adaptive self-organizing fuzzy neural network will be shown in section 2.3.

2.1 Growing Self-Organizing Algorithm (GA)

The growing self-organizing algorithm in the network model is used to confirm the fuzzy rules of the fuzzy neural network (FNN). In fact, we will confirm the number of the nodes in the FNN, and the nodes can be changed real-time. The details of the algorithm are shown as follows. The whole neural network architecture of GA is shown in Fig. 2.

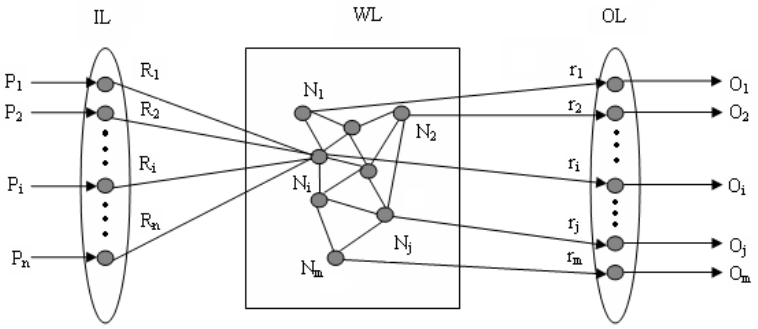


Fig. 2. The schematic of the growing self-organizing algorithm (GA)

where $P=\left((py_1,p^2y_1),(py_2,p^2y_2),...,(py_n,p^2y_n)\right)$ is the external vector, the number of nodes in the working layer (WL) are based on the input layer (IL). The number of the output layer (OL) is the same as the WL.

The main steps of the Growing Self-Organizing Algorithm:

(1) Initialization, taking the number of nodes $N\_num=1$, N_num represents the output nodes number, w represents weight value vector quantities, and w_1 is zero vector; the max time constant T and time $t=0$; parameters σ_o, k_σ, α_0, κ_α.

(2) Changing the data space P into $\overline{P}$.

$$\overline{py}_i = py_i / \sum_{j=1}^{n} py_j , \tag{1}$$

$$\overline{p^2 y}_i = p^2 y_i / \sum_{j=1}^{n} p^2 y_j , \tag{2}$$

$\overline{P}=[(\overline{py}_1,\overline{p^2 y}_1),(\overline{py}_2,\overline{p^2 y}_2),\cdots,(\overline{py}_n,\overline{p^2 y}_n)]$

(3) Using $\overline{P}$ as the input data space and calculating

$$G(\overline{p}) = \left\| \overline{p} - w_i^T \right\|, \ i=1,2,\cdots,n , \tag{3}$$

where w_i^T is the weight vector of node i.

(4) Judging to add a new node or not according to the following rule:

$$\begin{cases} G(\bar{p}) > \varphi(\bar{p}_i), N\_num+1 \\ G(\bar{p}) \le \varphi(\bar{p}_i), N\_num \end{cases}, \ i = 1,2,\cdots,n, \tag{4}$$

where $\varphi(\bar{p}_i) = 10^{(-4)} \times r \times e^{(r-3)}$; the value of r can be changed by the real system. If it needs to add a new node, $p = \bar{p}_i$, $N\_num = N\_num+1$. Feed forward coupling weight value $w_{N\_num} = p^T$; and then go to step (5). But if $N\_num = N\_num$, go to step (3).

(5) Calculating feed back coupling weight value near node p_t based on :

$$w_{tj} = \exp(-\frac{G(p^T)}{2\sigma(t)^2})(j = 1,2,\cdots,n), \tag{5}$$

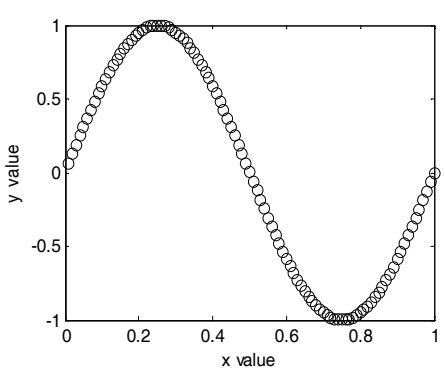
where $\sigma(t) = \sigma_0 \exp(-k_\sigma t)$; and calculating the feed forward weight value such as $w_j(t+1)(j = 1,2,\cdots,n)$ based on the adaptation of SOM.

(6) In order to optimize the neural network, the redundant node need to be reduced. The second training is to reduce the redundant nodes. The judgment is that whether every winner node has enough near nodes. The winner node which is a marginal node without near node will be reduced.

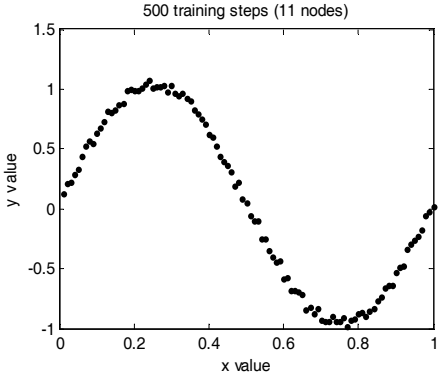
(7) The network stops calculating when every rule is satisfied.

When the number of nodes is confirmed, the network structure is also formed. But this kind of network structure based on the real-time system and can be changed real-time.

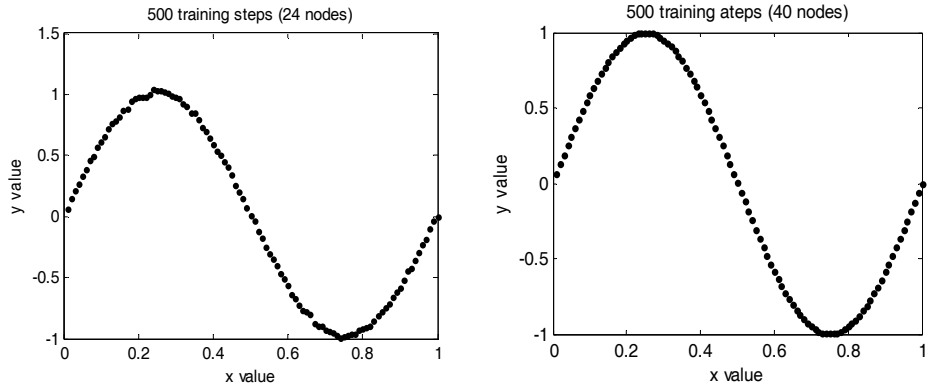
And we use this network to track the sine function: $y = \sin(2\pi x)$. The tracking results are shown in Fig. 3. Fig (a) is the restrict figure of the sine function; Fig (b) is the network with 11 nodes and within 500 training steps; Fig (c) is the network with 24 nodes within 500 training steps; Fig (d) is the figure with 40 nodes within 500 training steps. Here the parameters are initiated: $\sigma_o = 0.01$, $k_\sigma = -0.001$, $\alpha_0 = 0$, $k_\alpha = 0.01$.



(a)



(b)



(c) **(d)**

Fig. 3. The results of the sine function

The results of the Fig (b), (c) and (d) showed the process of the growing self-organizing algorithm. They proved that this growing self-organizing algorithm can change the weight value on-line, and we can find that when there were 40 nodes the algorithm can approach the sine function gradually. And the algorithm was not decided in advance. In the section 2.3 we will associate the growing self-organizing with the fuzzy neural network. The growing self-organizing fuzzy neural network can be used as a controller. The GSFNN controller will be laid in section 3. And we use the controller to control the dissolved oxygenic in the wastewater treatment.

2.2 Fuzzy Neural Network

In this section, we choose a kind of Fuzzy Neural Network (FNN) which consists of four layers. There are input layer, two hidden layers and an output layer. The structure of the neural network is shown in Fig. 4.

Every function of the neural network is such as:

First layer: Input Layer

There are two nodes this layer, which separately represent the error e and its derivative ec.

$$In_i^{(1)} = x_i \,,\; Out_i^{(1)} = In_i^{(1)} \,,\; (i = 1,2) \,. \tag{6}$$

Second layer: Fuzzification layer

The input parameter will be fuzzyfied. Each node here represents a language discussion of the input variable. We initiate to divide e into 2 parts and ec into 2 parts separately. So this layer has 4 nodes.

$$In_{ij}^{(2)} = Out_i^{(1)} \,,\; Out_{ij}^{(2)} = \mu(x_i) = e^{-(x_i - a_{ij})/b_{ij}} \,, \quad i = 1, j = 1,2 \,; i = 2, j = 1,2 \,. \tag{7}$$

In this layer the growing clustering algorithm based on GSOM will be used to change the number of the nodes. Where a are the membership functions of e, and b are the membership functions of ec. The algorithm will be discussed in section 3.

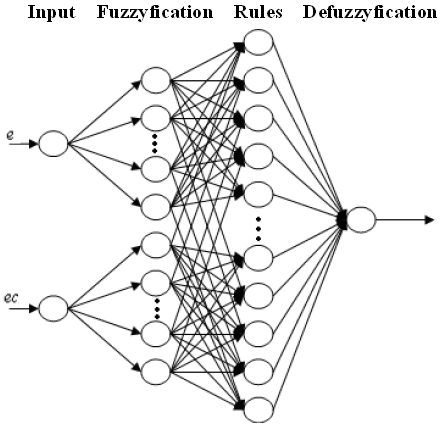


Fig. 4. The structure of the fuzzy neural network

Third layer: rules layer
Every neural cell in this layer represents an existing rule. We combine inputs with each other absolutely then attain expectation nodes. We calculate the fitness value of rule by taking the method of product in this layer.

$$In_{pq}^{(3)} = \mu_{1p}(x_i) \times \mu_{2q}(x_2),\ Out_{pq}^{(3)} = In_{pq}^{(3)}, \quad p = 1,2,\cdots,n;\ q = 1,2,\cdots,m. \tag{8}$$

The clustering algorithm will be used in this layer to ensure the nodes of the structure. Firstly, we also begin with $n = 2, m = 2$. Then new nodes will be inserted when the systems need.
Forth layer: Defuzzification layer
We clarify the output by gravity method. w_{pq} are the centers of the output languages discussion.

$$In^{(4)} = \sum_{p=1}^{n}\sum_{q=1}^{m}(Out_{pq}^{(3)} \times w_{pq}), \quad Out^{(4)} = \sum_{j=1}^{N}\omega_j^4 In^{(4)}, \quad p = 1,2,\cdots,n;\ q = 1,2,\cdots,m. \tag{9}$$

Initially, $n = 2, m = 2$.

This structure of the fuzzy neural network contains fewer layers than the conventional networks which usually with five or more layers. So this network is simple and can save much time by computing.

2.3 Adaptive growing Self-Organizing Fuzzy Neural Network

In this section the adaptive growing self-organizing fuzzy neural network will be shown. In the growing fuzzy neural algorithm, the GA method is used in the second layer and the third layer. And the outputs of the third layer are the fuzzy rules, but in order to make the main algorithm in this chapter be faster, it needs to change the value of the parameters in this layer. And then the rules can catch the requirements of the real systems. The process diagram of the growing fuzzy neural is draw as Fig5:

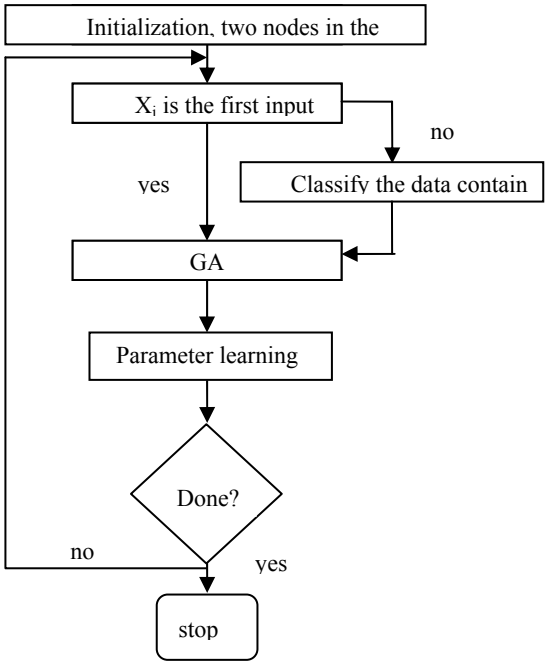


Fig. 5. The process diagram of the growing fuzzy neural algorithm

From the process diagram, the network nodes can be certain by the GA. It can be improved if a "default" rule is used in addition to the "normal" fuzzy rules. But in order to make the algorithm be convergent, there must be do parameter learning. The user can stop the addition of the fuzzy rules when the desired accuracy is reached.

3. The GSFNN Controller (GSFNNC)

Ferrer and his partners used fuzzy algorithm to control the dissolved oxygenic in the wastewater treatment (Ferrer J, Redrigo M A, Seco A, et a1, 1998). Syu and Chen used the BP

neural network to control the active sludge age in the wastewater treatment (Syu M J. Chert B C, 1998). In this chapter, we combine the advantages of these algorithm and use the fuzzy neural network to control the dissolved oxygenic in the wastewater treatment. In this network the nodes of the fuzzy nodes can be changed by the growing self-organizing algorithm. The schematic of the control system is shown in Fig. 6:

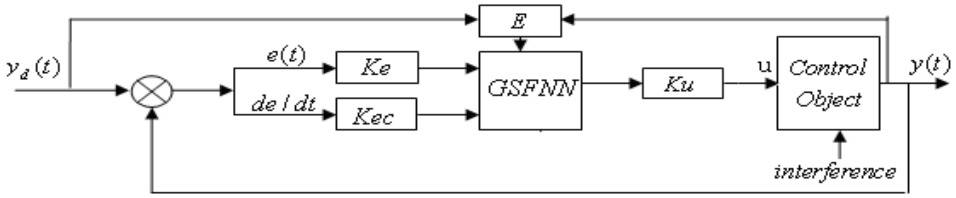


Fig. 6. The schematic diagram of control system

In Fig. 6 Ke and Kec are the quantifications of the controller input, Ku is the quantification of the controller output. GSFNN is the growing self-organizing fuzzy neural network controller. This controller owns the same functions as the traditional fuzzy controller: input, fuzzication, inferences, defuzzication. But this controller can change the nodes of the network, so it can change the structure of the network, realize the control rules real-time.

4. Simulations

Evidently, mathematical models are important not only for optimizing design but also for improving operation and control of these complex biological processes in wastewater treatment plants. Much of the work done on activated sludge bioreactor modeling in recent years has been largely concentrated on improving the understanding of the kinetics of the process (Talat Mahmood, Allan Elliott, 2006). The fundamental scheme of an activated sludge process of the wastewater treatment plant described in this chapter is shown in Fig.7.

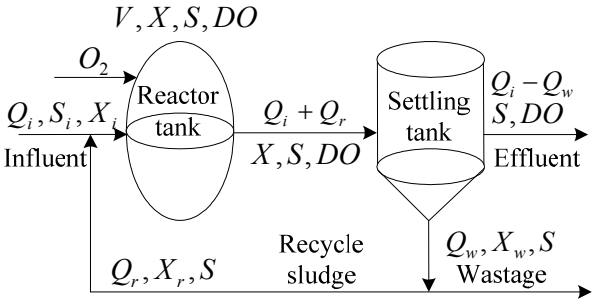


Fig. 7. The fundamental scheme of activated sludge process

The activated sludge process mainly includes the aerator system and the activated sludge recycling system. The process is a continuous system in which aerobic biological growths are mixed and aerated with wastewater in the reactor tank and separated in the settling tank. The aerobic biological growths get rid of organic matter included in wastewater.

In order to obtain the respective mathematical model, the following facts are assumed (Edgar N. Sanchez, Jose M. Gonzalez, Esperanza Ramirez, 2000):
1) The microorganisms' growth rate is bigger than their death rate and obeys the Monod law;
2) No biochemical reactions take place inside the settling tank;
3) Biomass in the sedimentation tank is negligible;
4) The inflow stream contains no biomass;
5) Complete settling is achieved, hence the sludge wastage is restricted to waste stream.
By means of mass balance for biomass concentration, substrate concentration and oxygen concentration, we obtain the following model:
A typical model of active sludge wastewater treatment is such as the following (FENG Yu-zhao, LONG Teng-rui, GUO Jing-song, et al., 2003):

$$\begin{cases} \dfrac{dx}{dt} = \bar{\mu}\dfrac{sx}{k_s + s} - k_d x + \dfrac{Q}{V}x_i - \dfrac{Q_w}{V}Cx \\ \dfrac{ds}{dt} = \dfrac{\bar{\mu}}{Y_{SH}}\dfrac{sx}{k_s + s} + \dfrac{Q}{V}(s_i - s) \\ \dfrac{do}{dt} = \dfrac{(1 - ff_s Y_{SH})}{fY_{SH}}\dfrac{sx}{k_s + s} - f_s k_d x + \mu \end{cases}, \tag{10}$$

where x is the biomass concentration in the aeration tank, x_i is the inflow biomass concentration, s is the reactor substrate concentration, s_i is the substrate concentration contained in inflow, o is the oxygen concentration in reactor tank, $\bar{\mu}$ is the maximal specific growth rate, k_s is the half-velocity constant, substrate concentration at one-half the maximal growth rate, k_o is the oxygen half-saturation coefficient for heterotrophic biomass, k_d is the endogenous decay coefficient, Q is the inflow, V is the reactor volume, Q_w is the wastage flow, C is the factor of concentration in the sedimentation tank (2), Y is the yield coefficient, Y_{obs} is the observed yield coefficient, f is a factor, which correlates substrate with oxygen demand, f_x is the consumption factor, and u is the oxygen transfer rate with BOD_5 standing for biochemical demand of oxygen during 5 days, VSS for volatile suspended solids, and COD for chemical demand of oxygen.Because of the character of the wastewater treatment, the paper (FENG Yu-zhao, LONG Teng-rui, GUO Jing-song, et al., 2003) gave the dynamic parameter of the process. The limitations of the parameters are given as follow.

$\bar{\mu} - k_d - CQ_w / V \in [2.9495, 5.9495]$;

$\bar{\mu} - Y_{NH} \in [9.0634, 18.1296]$;

$Q / V \in [6.05, 10.05]$;

$Q_{Si} / V \in [0.012, 0.142]$;

$$\frac{\bar{\mu}(1-ff_s Y_{NH})-ff_s Y_{NH} K_d}{fY_{NH}} \in [4.936, 9.943]$$

In this chapter, we choose a group numbers within these limitations, and one of the models will be obtained.

$$\begin{bmatrix} \dot{x}(t) \\ \dot{s}(t) \\ \dot{o}(t) \end{bmatrix} = \begin{bmatrix} 3.95 & 0 & 1 \\ 15.05 & -8.05 & 0 \\ 7.05 & 0 & -1 \end{bmatrix} \begin{bmatrix} x(t) \\ s(t) \\ o(t) \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} u(t) \\ q(t) \end{bmatrix} \tag{11}$$

$$y(t) = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x(t) \\ s(t) \\ o(t) \end{bmatrix} \tag{12}$$

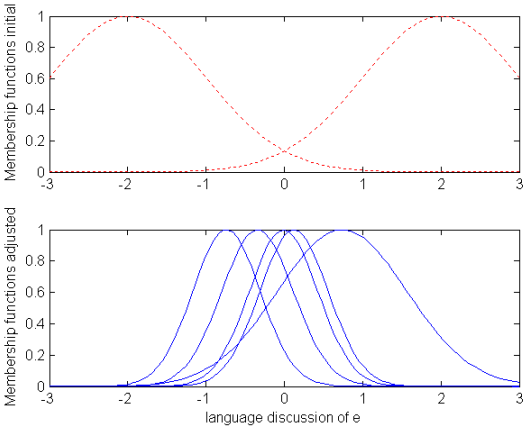
The dissolved oxygenic concentration of the wastewater treatment always contains about from 1.6 mg/L to 2.4 mg/L; in the simulations we use the dissolved oxygenic concentration near this requirement. In the traditional fuzzy neural network, the region value and the number of the nodes were ensured by the experience or experimentations. But these methods may be not applicable in the real-time systems. Besides, the control results of the simulations by the traditional fuzzy neural network with 7×7 fuzzy rules (or neural nodes) are nearly the same as the fuzzy neural network with 9×9 fuzzy rules. But the time of the experiments was not the same. The later was nearly twice as the former. It is unsuitable in the control systems. But the number of the fuzzy neural nodes can be ensured by the adaptive growing self-organizing algorithm. So the adaptive growing self-organizing fuzzy neural network controller can be used to control the systems without the experience.

Based on the neural network controller in the section 3, and the parameter of the dissolved oxygenic of the wastewater treatment system in the former of this section, we do some researches.

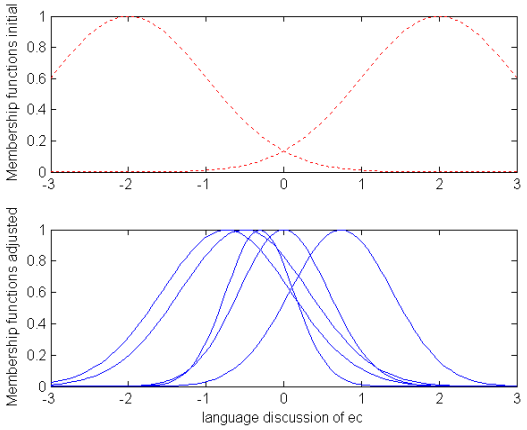
In the conventional fuzzy neural network, the region value and the number of the nodes were ensured by the experience or experimentations. And in this chapter we choose 49(7×7) fuzzy rules for the conventional fuzzy neural network. The growing self-organizing fuzzy neural network was initially given 4(2×2) fuzzy rules. After learning the number of the fuzzy rules can add to adapt the requirement of the system. The membership functions of linguistic variable e and ec are shown as Fig 8.

The results of the simulations are shown in Fig.9. In the process of wastewater treatment, the system requires dissolved oxygenic concentration to be 1.6 mg/L, and then 2.0 mg/L. Fig(a) shows the result of conventional fuzzy neural control by 49(7×7) fuzzy rules; Fig(b) shows the result of growing self-organizing fuzzy neural control by 25(5×5) fuzzy rules. We can find that the growing self-organizing fuzzy neural algorithm is faster than the conventional fuzzy neural algorithm by the same required error. Sometimes, the fuzzy rules need to be concerted by experience, the parameters of the fuzzy neural network need to be trained off-line. But we know from the former, the growing self-organizing fuzzy neural algorithm

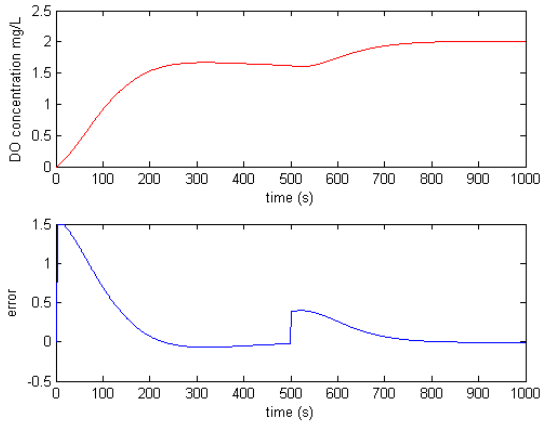
concerts the fuzzy rules by the real system, and trains the parameters on-line. Because we introduce the dynamic descent gradient method, the learning speed is faster than the conventional fuzzy neural algorithm. Besides, the storage space of the growing self-organizing fuzzy neural algorithm is about 101424 bytes. But the conventional fuzzy neural algorithm is about 298432 bytes. This is very useful in industry. So the new algorithm proposed in this chapter is more superiority than the conventional fuzzy neural algorithm.
Fig. 9 provides the growing self-organizing fuzzy neural network is faster and smoother than the traditional fuzzy neural network, and the Fig. 9 (b) provides the new method is robustness when applied to wastewater treatment system that has strong disturbances, and has strong self-adaptability when changing the requirement of dissolved oxygen (DO) concentration.



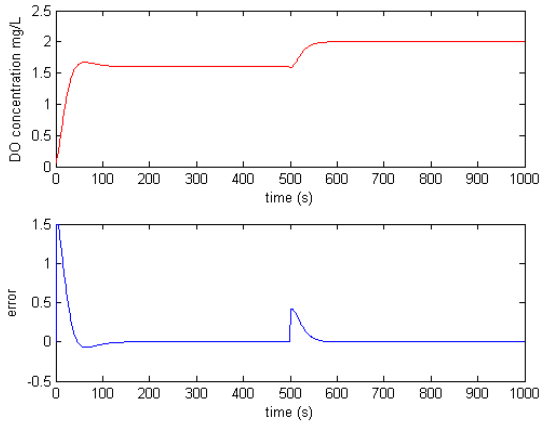
(a) Membership functions of *e* initial and after adjusted



(b) Membership functions of *ec* initial and after adjusted
Fig.8. Membership functions of inputs



(a) The result of conventional fuzzy neural control



(b) The result of growing self-organizing fuzzy neural control
Fig. 9. The results of the simulations

5. Conclusions

To summarize, the Adaptive Growing Self-Organizing Fuzzy Neural Network controller in this chapter solves the limitations of the traditional fuzzy neural network. The number of the neural network can be changed by the real systems. This chapter provides a new approach to the research of wastewater treatment process, and GSFNNC presents a good respond to a variety of conditions and offers the advantages of certain degree of robustness over the changes that the process can undergo. The simulation results have demonstrated that GSFNNC has abilities as follows:

1. The growing self-organizing algorithm can ensure the nodes' number by real systems;
2. This network controller can keep dissolved oxygen concentration at a proper level, avoiding sludge swelling;

3. This network controller can adjust membership function on-line, optimize control rules, and has strong robustness. It can work well under the condition of continuous influent and makes the effluent meet the discharge standard.

6. Referring

This work was supported by the National 863 Scheme Foundation of China under Grant 2007AA04Z160; National Science Foundation of China under Grant 60674066; National Science Foundation of China under Grant 60873043; Ph.D. Programs Foundation of Ministry of Education of China under Grant 200800050004; Beijing Municipal Natural Science Foundation under Grant 4092010.

7. References

T.Poggio and F.Girosi (1990). Networks for approximation and learning. *Proceedings of the IEEE*, pp.1481-1497.

Yu Zhao, Huijun Gao & Shaoshuai Mou (2008). Asymptotic stability analysis of neural networks with successive time delay components. *Neurocomputing*, Vol. 71, No. 13-15, pp. 2848–2856.

Fritzke, B. (1994). Growing cell structure—A self-organizing neural network for unsupervised and supervised learning. *Neural Networks*, Vol. 7, No. 9, pp. 1441–1460.

Kohonen T. (1982). Self-Organizing Formation of Topologically Correct Feature Maps. *Biol Cyber*, Vol. 43, pp. 59–69.

Dittenbach. M., Merkel. D & Rauber. A. (2000). The growing hierarchical self-organizing map. *Proceedings of the international joint conference on neural networks*, vol. VI, pp. 15–19.

Liying Ma and K. Khorasani (2005). Constructive Feed-forward Neural Networks Using Hermite Polynomial Activation Functions. *IEEE Trans. Neural Networks*, Vol. 16, No. 4, pp. 821–834.

R. Sentiono (2001). Feedforward neural network construction using cross validation, *Neural Computation*, Vol. 13, No. 12, pp. 2865–2877.

S. S. Ge, F. Hong, and T. H. Lee (2003). Adaptive neural control of nonlinear time-delay systems with unknown virtual control coefficients. *IEEE Trans. Syst., Man, Cybern. B*, Vol. 34, No. 1, pp. 499–516.

Junfei Qiao, Honggui Han, and Yanmei Jia (2007). An Adaptive Growing self-organizing fuzzy neural network. *The 5th International Conference on Wavelet Analysis and Pattern Recognition2007*, pp. 711–715.

Ferrer J, Redrigo M A, Seco A, et a1 (1998). Energy saving in the aeration process by fuzzy logic control . *Water Science and Technology*, Vol. 38, No. 3, pp. 209–217.

Syu M J. Chert B C. (1998). Back-propagation neural network adaptive control of a continuous wastewater treatment process. *Industrial engineering Chemistry Research*, Vol. 37, No.12, pp. 3625–3630.

Talat Mahmood, Allan Elliott (2006). A review of secondary sludge reduction technologies for the pulp and paper industry. *Water Research*, Vol. 40, No. 11, pp. 2093–2112.

Edgar N. Sanchez, Jose M. Gonzalez, Esperanza Ramirez (2000). Minimal PD fuzzy control of a wastewater treatment plant. *Proceedings of the 15th IEEE International Symposium on Intelligent Control*, pp. 169-173.

FENG Yu-zhao, LONG Teng-rui, GUO Jing-song, et al (2003). Optimal robustness control method of activated sludge system based on uncertain parameters,China Water & Wastewater, Vol. 19, No. 3, pp. 14-16.

2

Learning the number of clusters in Self Organizing Map

Guénaël Cabanes and Younès Bennani
University of Paris 13, LIPN-CNRS
France

1. Introduction

The Self-Organizing Map (SOM: Kohonen (1984, 2001)) is a neuro-computational algorithm to map high-dimensional data to a two-dimensional space through a competitive and unsupervised learning process. Self-Organizing Maps differ from other artificial neural networks in the sense that they use a neighborhood function to preserve the topological properties of the input space. This unsupervised learning algorithm is a popular nonlinear technique for dimensionality reduction and data visualization.
The SOM is often used as a first phase for unsupervised classification (i.e. clustering). Clustering methods are able to perform an automatic detection of relevant sub-groups or clusters in unlabeled data sets, when one does not have prior knowledge about the hidden structure of these data. Patterns in the same cluster should be similar to each other, while patterns in different clusters should not (internal homogeneity and the external separation). Clustering plays an indispensable role for understanding various phenomena described by data sets. A clustering problem can be defined as the task of partitioning a set of objects into a collection of mutually disjoint subsets. Clustering is a segmentation problem which is considered as one of the most challenging problems in unsupervised learning. Various approaches have been proposed to solve the problem (Jain & Dubes, 1988).
An efficient method to grouping problems is based on the learning of a Self-Organizing Map. In the first phase of the process, the standard SOM approach is used to compute a set of reference vectors (prototypes) representing local means of the data. In the second phase, the obtained prototypes are grouped to form the final partitioning using a traditional clustering method (e.g. K-means or hierarchical methods). Such an approach is called a two-level clustering method. In this work, we focus particular attention on two-level clustering algorithms. One of the most crucial questions in many real-world cluster applications is how to determine a suitable number of clusters K, also known as the model selection problem. Without a priori knowledge there is no simple way of knowing that number. The purpose of our work is to provide a simultaneous two-level clustering approach using SOM, by learning at the same time the structure of the data and its segmentation, using both distance and density information. This new clustering algorithm assumes that a cluster is a dense region of objects surrounded by a region of low density (Yue et al., 2004; Ultsch, 2005; Ocsa et al., 2007; Pamudurthy et al., 2007). This approach is very effective when the clusters are

irregular or intertwined, and when noise and outliers are present. The proposed clustering algorithm divides automatically a given dataset into a collection of subsets (clusters), i.e., the number of clusters is determined automatically during the learning process, i.e., no a priori hypothesis for the number of clusters is required. This approach has been tested on a set of critical clustering problems and shows excellent results compared to usual approaches.
The remainder of this chapter is organized as follows. Section 2 presents the DS2L-SOM algorithm (Density-based Simultaneous Two-Level - SOM). Section 3 describes the validation databases and experimental protocol. In section 4 we show validation results and their evaluation. Conclusion and future work perspectives are given in Section 5.

2. Local Density-based Simultaneous Two-Level Clustering

High dimension data may be sparse (the curse of dimensionality), making it difficult for a clustering algorithm to find any structure in the data. Indeed, when dimensionality increases, data become increasingly sparse. Definitions of density and distance between objects, which is critical for clustering and outliers detection, become less meaningful. To improve the solution for this problem, a large number of dimension reduction approaches have been developed and tested in different application domains and research communities. The main idea behind these techniques is to map each pattern into a lower dimensional space that preserves the topology of data. The reduced data present at the lower dimensional representation can be used to perform clustering more efficiently. Various approaches have been proposed for the two-level clustering problem (Aupetit, 2005; Bohez, 1998; Hussin et al., 2004; Ultsch, 2005; Korkmaz, 2006). The key idea of the two-level clustering approach based on SOM is to combine the dimensionality reduction and the fast learning capabilities of SOM in the first level to construct a new reduced vector space. Then another clustering method is applied in this new space to produce a final set of clusters at the second level (Hussin et al., 2004; Ultsch, 2005). Although the two-level methods are more interesting than the traditional approaches, the data segmentation obtained from the SOM is not optimal, since a part of information is lost during the first stage (dimensionality reduction).
We propose here a new unsupervised learning algorithm (DS2L-SOM) which learns simultaneously the structure of the data and its segmentation using both distance and density information.

2.1 Principle

Kohonen SOM (Kohonen, 1984, 2001) can be defined as a competitive unsupervised learning neural network. When an observation is recognized, the activation of an output cell - competition layer - inhibits the activation of other neurons and reinforces itself. It is said that it follows the so called "Winner Takes All" rule. Actually, neurons are specialized in the recognition of one kind of observation. A SOM consists in a two dimensional map of neurons which are connected to n inputs according to n weights connections $w^{(i)}=(w_0^{(i)},\ldots,w_n^{(i)})$ and to their neighbors with topological links. The training set is used to organize these maps under topological constraints of the input space. Thus, a mapping between the input space and the network space is constructed; two close observations in the input space would activate two close units of the SOM. An optimal spatial organization is determined by the SOM from the input data, and when the dimension of the input space is

lower than three, both position of weights vectors and direct neighborhood relations between cells can be represented visually. Thus, a visual inspection of the map provides qualitative information about the map and the choice of its architecture. The winner neuron updates its prototype vector, making it more sensitive for later presentation of that type of input. This allows different cells to be trained for different types of data. To achieve a topological mapping, the neighbors of the winner neuron can adjust their prototype vector towards the input vector as well, but at a lesser degree, depending on how far away they are from the winner. Usually a radial symmetric Gaussian neighborhood function K_{ij} is used for this purpose.

2.2 DS2L-SOM algorithm

Connectionist learning algorithms are often presented as a minimization of a cost function. In our case, it will correspond to the minimization of the distance between the input samples and the map prototypes, weighted by a neighborhood function K_{ij}. To do that, we use a gradient algorithm. The cost function to be minimized is defined by:

$$R(w) = \frac{1}{N} \sum_{k=1}^{N} \sum_{j=1}^{M} K_{j,u^*(x^{(k)})} \parallel w^{(j)} - x^{(k)} \parallel^2$$

N represents the number of learning samples, M the number of neurons in the map, $u^*(x^{(k)})$ is the index of the neuron whose weight vector is the closest to the input pattern $x^{(k)}$ (the best match unit: BMU), and K_{ij} is a positive symmetric kernel function: the neighborhood function. The relative importance of a neuron i compared to a neuron j is weighted by the value of the kernel function K_{ij} which can be defined as:

$$K_{i,j} = \frac{1}{\lambda(t)} \times e^{-\frac{d_1^2(i,j)}{\lambda^2(t)}}$$

$\lambda(t)$ is the temperature function modeling the topological neighborhood extent, defined as:

$$\lambda(t) = \lambda_i \left(\frac{\lambda_f}{\lambda_i}\right)^{\frac{t}{t_{max}}}$$

λ_i and λ_f are respectively the initial and the final temperature (for example $\lambda_i = 2$, $\lambda_f = 0.5$). t_{max} is the maximum number allotted to the time (number of iterations for the x learning sample). $d_1(i,j)$ is the Manhattan distance defined between two neurons i and j on the map grid, with the coordinates (k,m) and (r,s) respectively:

$$d_1(i,j) = | r - k | + | s - m |$$

The DS2L-SOM algorithm is an adaptation of the S2L-SOM algorithm (Cabanes & Bennani, 2007). In S2L-SOM, each neighborhood connection is associated with a real value v which indicates the relevance of the connected neurons. The value of this connection is adapted

during learning process. It was proved by Martinetz (Martinetz, 1993) that the so generated graph is optimally topology-preserving in a very general sense. In particular each edge of this graph belongs to the Delaunay triangulation corresponding to the given set of reference vectors. For each data, both best close prototypes are linked by a topological connection. The value of this connection will be increased, whereas the value of all other connections from the best match unit will be reduced. Thus, at the end of the training, the set of inter-connected prototypes will be an artificial image of well separated sub-groups of the whole data set. Indeed, S2L-SOM can only detect borders defined by large inter cluster distances. However, the core part of a cluster can be defined as a region with high density and, in most cases, the cluster borders are defined either by an "empty" region between clusters (i.e. large inter-cluster distances) or by a low density region (Ultsch, 2005). In the DS2L-SOM algorithm, we propose also to associate each unit i to an estimate of the local data density $D^{(i)}$, so as to detect local fluctuations of density, which define the borders of touching clusters (low density regions). For each data, this density value will be increased for all units, as a function of the Euclidean distance between the related prototype $w^{(i)}$ and the data. This method of evaluation is similar to the one proposed by Pamudurthy et al. (2007). Silverman (1986) has shown that when the number of data points tends to infinity, the estimator D converges asymptotically to the true density function. One can notice that, in the DS2L-SOM algorithm, the estimation of the local density data is made during the training of the map, i.e. it is not necessary to keep the data in memory.

The DS2L-SOM learning algorithm proceeds essentially in three phases:

Input:

- Data $X = \{x^{(i)}\}_{i=1..N}$.
- SOM with M prototypes $\{w^{(i)}\}_{i=1..M}$.
- t_{max} : maximum number of iterations.

Output:

- A partition $P = \{C_i\}_{i=1..L}$ sets of inter-connected units.
- Density values $\{D^{(i)}\}_{i=1..M}$ associated to each unit.

1. Initialization phase:
   - Initialize all neighborhood connections values v to zero.
   - Initialize all unit density values $D^{(i)}$ to zero
2. Competition phase :
   - Present an input pattern $x^{(k)}$ to the SOM.
   - Choose the first BMU u* and the second BMU u**:

$$u^*(x) = \text{argmin}_{1 \le i \le M} \left\| x^{(k)} - w^{(i)} \right\|^2$$

$$u^{**}(x) = \text{argmin}_{i \ne u^*(x)} \left\| x^{(k)} - w^{(i)} \right\|^2$$

3. Adaptation phase:
   - Update prototype vectors $w^{(i)}$ of each unit i according to the learning rate $\varepsilon(t)$:

$$w^{(i)}(t) = w^{(i)}(t\text{-}1) - \varepsilon(t) K_{i,u^*(x^{(k)})}(w^{(i)}(t\text{-}1) - x^{(k)})$$

   - Increase local density value $D^{(i)}$ for each unit i :

$$D^{(i)}(t) = D^{(i)}(t-1) + r(t)e^{-\frac{\|x^{(k)} - w^{(i)}(t)\|^2}{2\lambda^2(t)}}$$

with

$$r(t) = \frac{1}{1 + e^{\left(-\frac{t}{t_{max}}\right)}}$$

- Let $\aleph(u^*)$ be the topological neighborhood of the first BMU; update the neighborhood connections values v according to the following rules:

$$v_{u^*,u^{**}}(t) = v_{u^*,u^{**}}(t\text{-}1) + r(t)|\aleph(u^*)|$$

and

$$v_{u^*,i}(t) = v_{u^*,i}(t\text{-}1) \text{ - } r(t)|\aleph(u^*)|$$
$$\forall i \in \aleph(u^*),\ i \neq u^{**}$$

where $|\aleph(u^*)|$ is the number of unit in $|\aleph(u^*)|$.

4. Repeat steps 2 and 3 until $t=t_{max}$.
5. Extract all clusters : Let $P = \{C_i\}_{i=1..L}$ the set of the L groups of linked units such as $v>0$ (see Fig.1(b)).
6. Return $P = \{C_i\}_{i=1..L}$.

At the end of the learning process, we use a refinement algorithm, which exploits connection and density information to detect clusters:

Input: P and $\{D^{(i)}\}_{i=1..M}$.

Output: The refined clusters.

1. For each $C_k \in P$ do:
   - Find the set $M(C_k)$ of density maxima (i.e. density mode, see Fig. 1(c)).

$$M(C_k) = \{\text{unit } i \in C_k \mid D^{(i)} \geq D^{(j)}, \forall j \in \aleph(i)\}$$

   - Determine the threshold matrix:

$$S = [S(i,j)]_{i,j:1\ldots|M(C_k)|}$$

with

$$S(i,j) = \left(\frac{1}{D^{(i)}} + \frac{1}{D^{(j)}}\right)^{-1}$$

   - For all unit $i \in C_k$, label the unit i with one element *label(i)* of $M(C_k)$, according to an ascending density gradient along the topological connections. Each label represents a sub-cluster (Fig. 1(d)).

- For each pair of neighbors unit (i,j) in C_k, if *label(i)* ≠ *label(j)* and if both $D^{(i)}>S(label(i),label(j))$ and $D^{(j)}>S(label(i),label(j))$ then merge the two sub-clusters (Fig. 1(e)).

2. Return refined clusters.

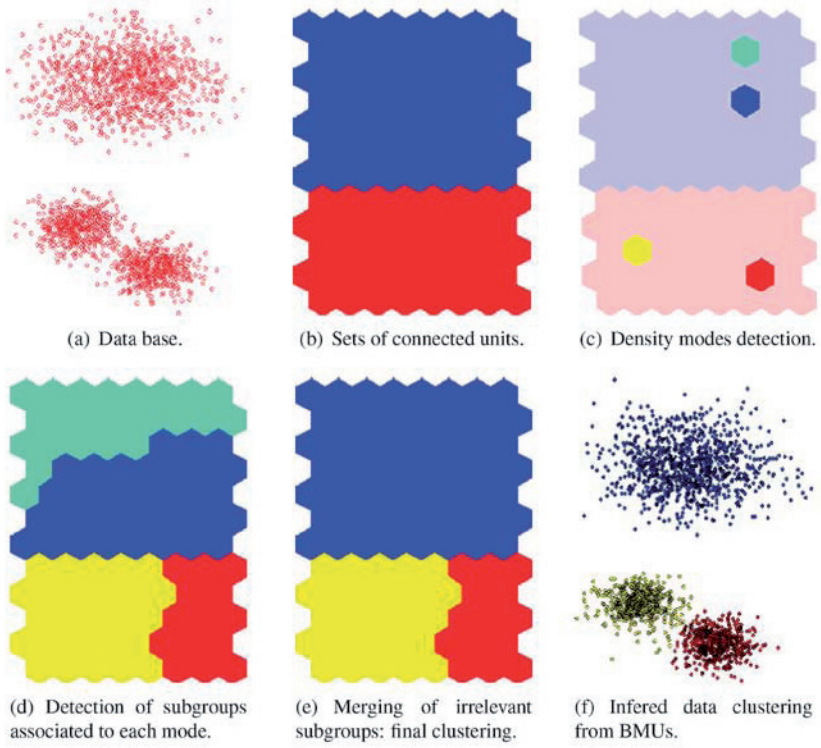


Fig. 1. Example of a sequence of the different stages of the refinement algorithm.

At the end of the learning process, the prototypes linked together by neighborhood connections having value v>0 define well separate clusters. Thus, we use a "Watersheds" method (see Vincent & Soille (1991)) on the density map of each of these clusters to find low density area inside well separate clusters, in order to characterize density defined sub-clusters. For each pair of adjacent subgroups we use a density-dependent index (Yue et al., 2004) to check if a low density area is a reliable indicator of the data structure, or whether it should be regarded as a random fluctuation in the density. This process is very fast because of the small number of prototypes. The combined use of these two types of group definition can achieve good results despite the low number of prototypes in the map. This allows different cells to be trained for different types of data. To achieve a topological mapping, the neighbors of the winner neuron can adjust their prototype vector towards the input vector as well, but at a lesser degree, depending on how far away they are from the winner. Usually a radial symmetric Gaussian neighborhood function K_{ij} is used for this purpose.

3. Experiments

3.1 Databases description

To demonstrate the effectiveness of the proposed two-level clustering method, the performances of the DS2L-SOM algorithm have been tested on 10 databases presenting various clustering difficulties (see Fig.2).

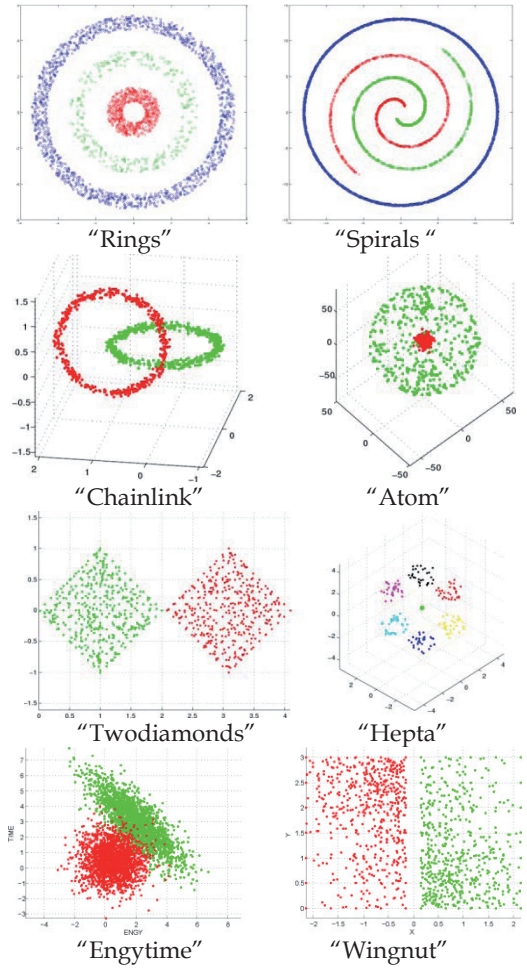


Fig. 2. Data visualizations.

The databases "Hepta", "Chainlink", "Atom", "Twodiamonds", "Engytime" and "Wingnut" come from the Fundamental Clustering Problem Suite (FCPS: Ultsch (2005)). We also generated four other interesting data bases ("Rings", "Spirals", "HighDim" and "Random"). "Rings" is made up of 3 groups in 2 dimensions not linearly separable with different densities and variances: a ring of radius 1 with 700 points (strong density), a ring of

radius 3 with 300 points (low density) and a ring of radius 5 with 1500 points (average density). "HighDim" consists of 9 quite separate groups of 100 points each one in a 15 dimensions space. "Random" is a random generation of 1000 points in 8 dimensions space. Finally "Spirals" consists of two parallel spirals of 1000 points each one in rings of 3000 points. The density of points in the spirals decreases with the radius.

3.2 Experimental protocol

We compared the performances of the DS2L-SOM algorithm, in term of segmentation quality and stability, to S2L-SOM (Cabanes & Bennani, 2007), and to the traditional two levels methods. The selected algorithms for comparison are K-means and SingleLinkage applied to the prototypes of the trained SOM. The Davies-Bouldin index (Davies & Bouldin, 1979) is used to determine the best cutting of the dendrogram (SingleLinkage) or the optimal number K of centroids for K-means. This index, suggested by Davies and Bouldin (Davies & Bouldin, 1979) for different values k (cluster number), is defined as in the following, in order to combine the concepts of cluster separation (denominator) and cluster compactness (numerator):

$$DB(k)=\frac{1}{K}\sum_{i=1}^{K}\frac{s_i+s_j}{\|c_i-c_j\|^2}$$

being s_i the square root of the average error (within-cluster variance) of cluster i with the centroid c_i. S2L-SOM and DS2L-SOM determine the number of clusters automatically and do not need to use this index.

For the single link hierarchical clustering, the proximity of two clusters is defined as the minimum of the distance between any two objects in the two different clusters. The single link technique is good at handling non-elliptical shapes, but is sensitive to noise and outliers.

In this paper the quality of the clustering has been evaluated using external criteria (Overlap Indices Rand and Jaccard) frequently used (Halkidi et al., 2001, 2002) :

$$\text{Rand} = \frac{a_{00}+a_{11}}{a_{00}+a_{01}+a_{10}+a_{11}}$$

$$\text{Jaccard} = \frac{a_{11}}{a_{01}+a_{10}+a_{11}}$$

Where a_{11} denotes the number of object pairs belonging to the same label and to the same cluster, a_{10} denotes the number of pairs that belong to the same label but different clusters, and a_{01} denotes the pairs in the same cluster but with different labels. Finally, a_{00} denotes the number of object pairs sharing neither label nor cluster.

Indeed, if data-independent labels (categories) are available, the question may be asked of how well a given cluster solution corresponds to these external labels.

The concept of cluster stability is also used as an indicator for assessing the validity of data partitioning found by different algorithms. In order to evaluate the stability of the various algorithms, we use sub-sampling based method for each data bases (Ben-Hur et al., 2002). We then compute the difference between two different segmentations using the Jaccard index. We repeat this procedure several times to averaging the repeated index values. This average is regarded as a reliable estimate of the clustering stability.

4. Results

The results for the external indices show that for all the databases DS2L-SOM is able to find without any error the expected data segmentation and the right number of clusters. This is not the case of the other algorithms, when the groups have an arbitrary form, or when there is no structure in the data or the groups are in contact (see Table 1 and Fig. 3).

Data	SSL	SKM	S2L-SOM	DS2L-SOM	**True**
HighDim	4	9	9	9	**9**
Chainlink	2	11	2	2	**2**
Atom	6	6	2	2	**2**
Twodiamonds	2	2	2	2	**2**
Rings	7	13	3	3	**3**
Spirals	12	10	3	3	**3**
Hepta	5	7	7	7	**7**
Random	10	15	1	1	**1**
Wingnut	6	11	1	2	**2**
Engytime	10	10	1	2	**2**

Table 1. Number of clusters obtained by various clustering methods (SSL = SOM+SingleLinkage, SKM = SOM+K-means).

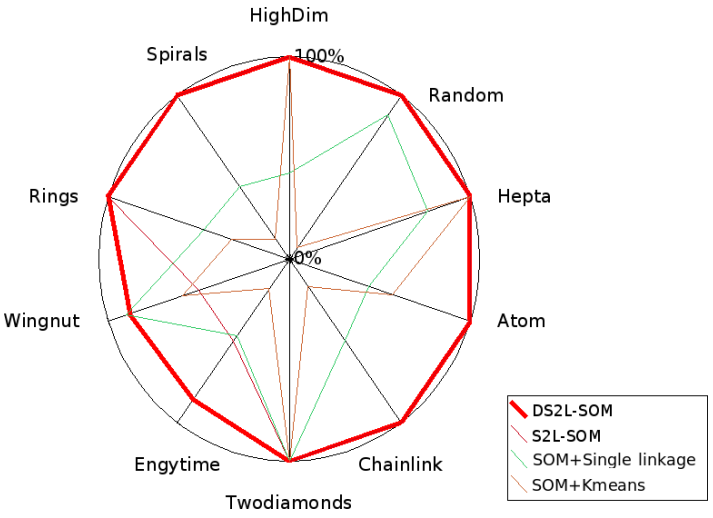


Fig. 3. Clustering quality using the Jaccard index for each algorithm on each database.

Considering the stability, the DS2L-SOM algorithm, like the S2L-SOM algorithm, shows excellent results for the data grouped in hyperspheric clusters, whatever the dimension ("Hepta" and "HighDim"), and also in the cases where the groups have arbitrary forms in two dimensions ("Rings" and "Spirals") and when the data are not structured ("Random"). It is worth noticing that in this last case the segmentation obtained by the most traditional methods is extremely unstable (Fig. 4).

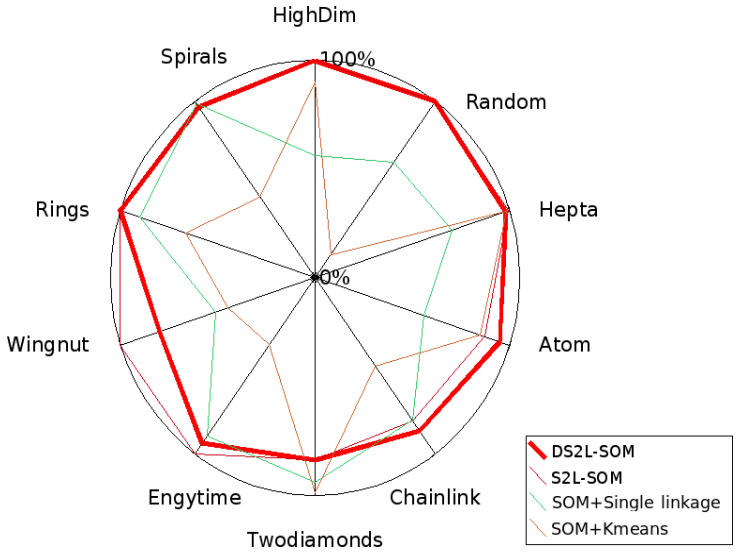


Fig. 4. Stability index for each algorithm on each database.

When the data are not linearly separable in dimensions higher than two ("Atom" and "Chainlink"), S2L-SOM and DS2L-SOM are limited by the topological constraint in two dimensions of the SOM network. Consequently, the stability of the segmentation is not maximum. However one can note that even in this case the DS2L-SOM algorithm remains more stable than the other methods. Moreover, when the clusters are defined only by the density ("Twodiamonds", "Engytime", "Wingnut"), sub-sampling may smooth the fluctuations of data density. This reduces the stability of the segmentation. In this case S2L-SOM is more stable than DS2L-SOM, because it can't separate this kind of groups and alway finds one group in each sub-sample.

The results are also confirmed by visual inspection. Indeed, the DS2L-SOM clustering algorithm is a powerful tool for visualization of the obtained segmentation in two dimensions. Clusters are easily and clearly identifiable, as well as regions without data (unconnected neurons). As one can notice it from figures 5 to 9, the results obtained by the DS2L-SOM algorithm are closer to reality than those found by the other algorithms. Figures 7 and 8 show that DS2L-SOM is able to detect density-defined clusters.

In these figures, each hexagon represents a prototype of the SOM together with its associated data. Hexagons showing the same color are in the same cluster. White hexagons are not part of any cluster.

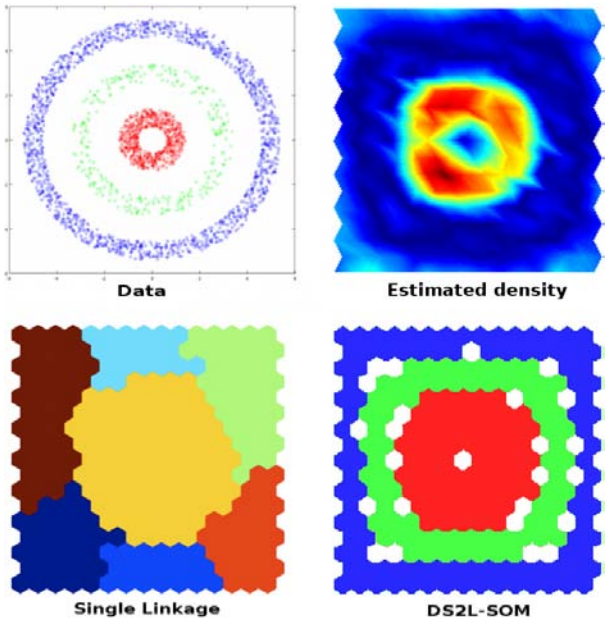


Fig. 5. Clustering of "Rings" data.

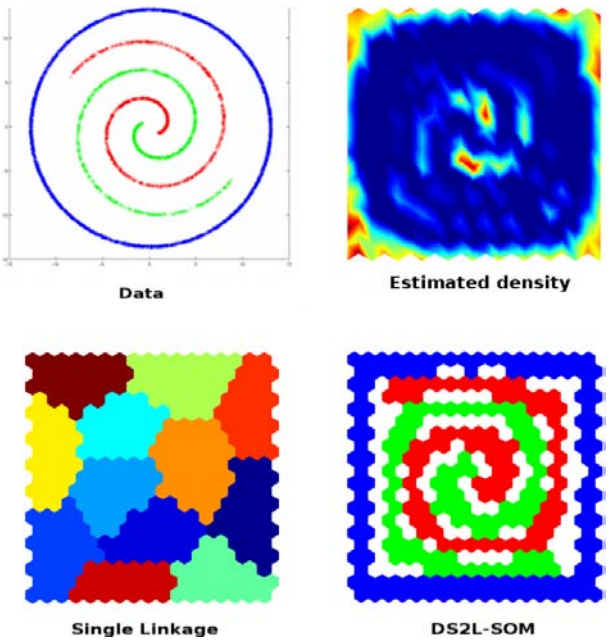


Fig. 6. Clustering of "Spirals" data.

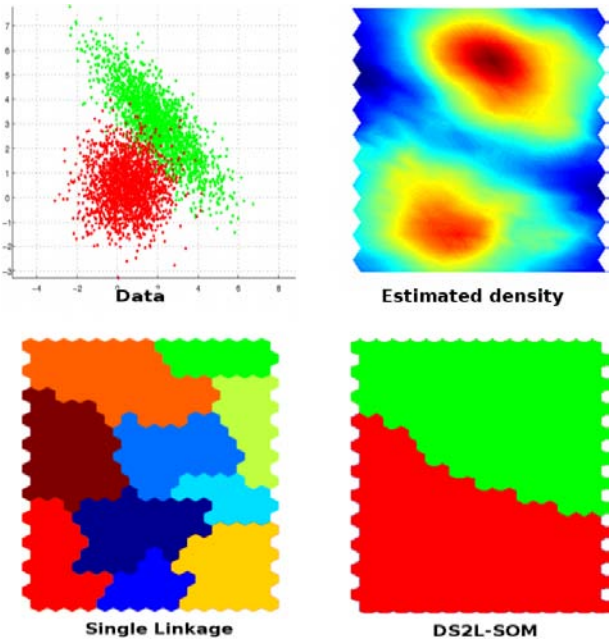


Fig. 7. Clustering of "Engytime" data.

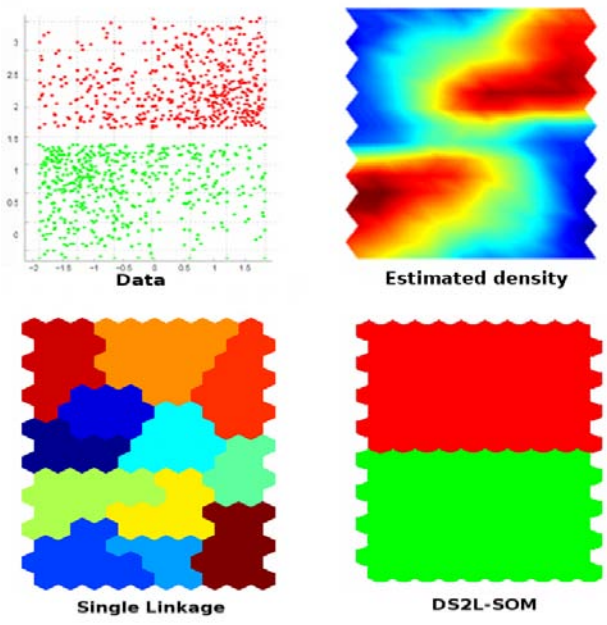


Fig. 8. Clustering of "Wingnut" data.

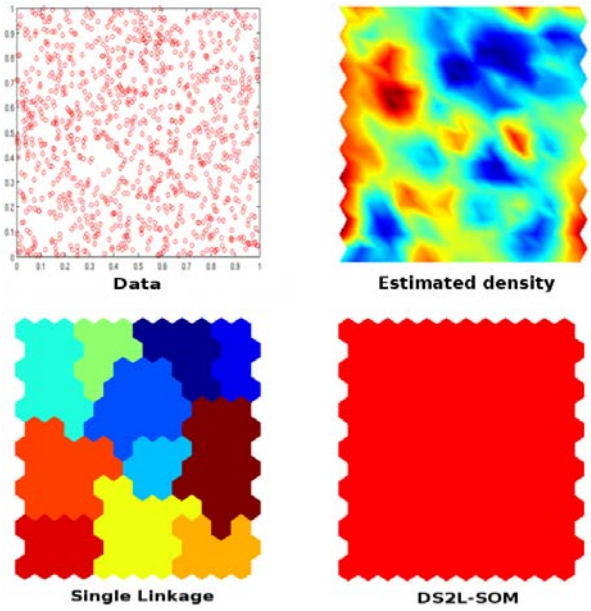


Fig. 9. Clustering of "Random" data. Data visualization is a two-dimensional projection of the "Random" database.

5. Conclusion

We proposed here a density-based simultaneous two-level clustering method. It uses SOM as dimensionality reduction technique and achieves an improved final clustering in the second level, using both distance and density information. The proposed algorithm DS2L-SOM locates regions of high density that are separated from one another by regions of low density. The performance of DS2L-SOM have been evaluated on a set of critical clustering problems, and compared to other two-level clustering algorithms. The experimental results demonstrate that the proposed clustering method achieves a better clustering quality than classical approaches. The results also demonstrate that DS2L-SOM is able to discover irregular and intertwined clusters, while conventional partitional clustering algorithms can deal with convex clusters only. Finally, the number of clusters in our approach is determined automatically during the learning process, i.e., no a priori hypothesis for the number of clusters is required. In the future we plan to incorporate in the DS2L-SOM algorithm some plasticity property, to evaluate its impact on the cluster quality and stability.

6. Acknowledgment

This work was supported in part by the Sillages project (N° ANR-05-BLAN-0177-01) financed by the ANR (Agence Nationale de la Recherche).

7. References

Aupetit, M. (2005). Learning topology with the generative gaussian graph and the EM algorithm. In *Neural Information Processing Systems (NIPS)* Vancouver, B.C., Canada.

Ben-Hur, A.; Elisseeff, A. & Guyon, I. (2002). A stability based method for discovering structure in clustered data. *Pacific Symposium on Biocomputing*, 7, 6–17.

Bohez, E. L. J. (1998). Two level cluster analysis based on fractal dimension and iteratedfunction systems (ifs) for speech signal recognition. *IEEE Asia-Pacific Conference on Circuits and Systems*, (pp. 291–294).

Cabanes, G. & Bennani, Y. (2007). A simultaneous two-level clustering algorithm for automatic model selection. In *Proceedings of the International Conference on Machine Learning and Applications (ICMLA'07)* Cincinnati, Ohio, USA.

Davies, D. L. & Bouldin, D. W. (1979). A cluster separation measure. *IEEE Transactions on Pattern Recognition and Machine Intelligence*, 1(2), 224–227.

Halkidi, M.; Batistakis, Y. & Vazirgiannis, M. (2001). On Clustering Validation Techniques. *Journal of Intelligent Information Systems*, 17(2-3), 107–145.

Halkidi, M.; Batistakis, Y. & Vazirgiannis, M. (2002). Cluster Validity Methods. *SIGMOD Record*, 31(2,3), 40–45, 19–27.

Hussin, M. F.; Kamel, M. S. & Nagi, M. H. (2004). An efficient two-level SOMART document clustering through dimensionality reduction. In *ICONIP* (pp. 158–165).

Jain, A. K. & Dubes, R. C. (1988). *Algorithms for clustering data*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc.

Kohonen, T. (1984). *Self-Organization and Associative Memory*. Berlin: Springer-Verlag.

Kohonen, T. (2001). *Self-Organizing Maps*. Berlin: Springer-Verlag.

Korkmaz, E. E. (2006). A two-level clustering method using linear linkage encoding. *International Conference on Parallel Problem Solving From Nature, Lecture Notes in Computer Science*, 4193, 681–690.

Martinetz, T. (1993). Competitive hebbian learning rule forms perfectly topology preserving maps. In S. Gielen & B. Kappen (Eds.), *Proceedings of the International Conference on Artificial Neural Networks (ICANN-93), Amsterdam* (pp. 427–434). Heidelberg: Springer.

Ocsa, A.; Bedregal, C. & Cuadros-Vargas, E. (2007). DB-GNG: A constructive self-organizing map based on density. *Prodeeding of International Joint Conference on Neural Networks*, (pp. 1506–1511).

Pamudurthy, S. R.; Chandrakala, S. & Sakhar, C. C. (2007). Local density estimation based clustering. *Prodeeding of International Joint Conference on Neural Networks*, (pp. 1338–1343).

Silverman, B. W. (1986). *Density Estimation for Statistics and Data Analysis*. Chapman & Hall/CRC.

Ultsch, A. (2005). Clustering with SOM: U*C. In *proc. workshop on self-organizing maps* (pp. 75–82). paris, france.

Vincent, L. & Soille, P. (1991). Watersheds in digital spaces: An efficient algorithm based on immersion simulation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 13, 583–598.

Yue, S.-H.; Li, P.; Guo, J.-D. & Zhou, S.-G. (2004). Using greedy algorithm: DBSCAN revisited II. *Journal of Zhejiang University SCIENCE*, 5(11), 1405–1412.

3

Improvements Quality of Kohonen Maps Using Dimension Reduction Methods

Jiří Dvorský, Václav Snášel, Jana Kočíbová
VŠB – Technical University of Ostrava
Czech Republic

1. Introduction

The performance of *Self Organizing Map* (SOM) is always influenced by learn methods. The resultant quality of the SOM is also highly dependent onto the learning rate and the neighborhood function. In literature, there are plenty of studies to find a proper method to improve the quality of learning process of the SOM. They focus especially on convergence (Cottrell et al., 1998; Kohonen, 2001), the measure of the global topology preservation (Bauer et al., 1999) and, at an individual level, the sensitivity to parameters such as initialization, rate of decrease of neighborhood function, optimum learning rate etc. (de Bodt & Cottrell, 2000; Mulier & Cherkassky, 1995; Germen, 2005; Flanagan, 1996; 1994).
Although various disciplines use the SOM model in order to find solutions to broad spectrum of problems, however, there is not so much clue about the how the resultant maps are supposed to look after training.
Most articles are focused on the learning process of the SOM. The quality of the SOM is needed to measure in this process. The question is how to measure this quality. The distortion, or distortion measure, is certainly the most popular criterion for assessing the quality of the classification of the SOM (Kohonen, 2001; Rynkiewicz, 2006). Distortion measure provides an assessment of SOM properties with respect to the data and overcomes the absence of cost function in the SOM algorithm. Usually *Mean Squared Error* (MSE) is used to measure a distortion. The MSE is just a number without any dimension or scale, and may be hard to understand. What is the value of distortion is small enough? At what point should be the learning process terminated? Alternative approach for measurement of quality of learning process is the goal of our research.
In the SOM each neuron represents a set of input vectors. As the learning process continues, the set should be more and more stable, i.e. particular input vector should not move from one set to another in successive iteration in learning process. Movements, which still occur, can measure the quality of the learning process or quality of resultant SOM, if we decide to stop the learning.
Another idea is usage of a dimension reduction methods to capture most significant features of resultant SOM (Dvorský, 2007). At the beginning of the learning process, weights in the SOM are initialized with random values. In this case, there is no common, important feature in the SOM – the SOM contains only noise. How learning process continues, the map will learn significant features in the data. These features should be dominant, and if some approximation of the SOM is computed, these features must be preserved. In this moment, SVD or HOSVD, see sections 2.1 and 2.2, can be used to compute SOM approximation. In terms of

SVD, dominant features of the SOM will be represented as greater singular values. And vice versa, the small singular numbers will represent some noise in the SOM only. If small values are neglected, the approximated SOM will be very similar to the original one. In this way, quality of the SOM can be measured using data dimension reduction methods.

1.1 Notation of the SOM

In this chapter we will consider SOM as an array of neurons $\mathbf{m_i}$, usually arranged in a low dimensionality grid (1D or 2D), the map, for ease of visualization. The grid may have several forms like rectangular, hexagonal. For each input vector $\mathbf{x}(t)$, neuron that best matches input vector is selected. This neuron is called *Best Matching Unit* (*BMU*). Then the weights of the BMU and its neighborhood will be adapted as follows:

$$\mathbf{m}_i(t+1) = \mathbf{m}_i(t) + \eta(t)h(t)(\mathbf{x}(t) - \mathbf{m}_i(t)), \quad \forall \mathbf{m}_i \tag{1}$$

where η, $0 < \eta < 1$ is the learning factor, which determines the speed of weight adaptation, and $h(t)$ is neighborhood size determining function.

2. Methods of Dimension Reduction

Since our approach is based on SVD and HOSVD techniques, we first briefly review matrix SVD and then introduce tensor and the HOSVD technique. In this chapter, tensors are denoted by calligraphic upper-case letters $(\mathcal{A}, \mathcal{B}, \ldots)$, matrices by uppercase letters $(A, B, \ldots)$, vectors by bold lower case letters $(\mathbf{a}, \mathbf{b}, \ldots)$, and scalars by lower case letters $(a, b, \ldots)$.

2.1 Singular Value Decomposition

Singular value decomposition (*SVD*) is well known because of its application in information retrieval – *Latent semantic indexing* (*LSI*) (Berry & Browne, 1999; Berry et al., 1995). SVD is especially suitable in its variant for sparse matrices (Larsen, 1998).

Theorem 2.1 (Singular value decomposition) *Let A is an $n \times m$ rank-r matrix. Be $\sigma_1 \geq \cdots \geq \sigma_r$ eigenvalues of a matrix $\sqrt{AA^T}$. Then there exist orthogonal matrices $U = (\mathbf{u}_1, \ldots, \mathbf{u}_r)$ and $V = (\mathbf{v}_1, \ldots, \mathbf{v}_r)$, whose column vectors are orthonormal, and a diagonal matrix $\Sigma = diag(\sigma_1, \ldots, \sigma_r)$. The decomposition $A = U\Sigma V^T$ is called* singular value decomposition *of matrix A and numbers $\sigma_1, \ldots, \sigma_r$ are* singular values *of the matrix A. Columns of U (or V) are called* left (or right) singular vectors *of matrix A.*

Now we have a decomposition of original matrix A. It is not needed to say, that the left and right singular vectors are not sparse. We have at most r nonzero singular numbers, where rank r is the smaller of the two matrix dimensions. Luckily, because the singular values usually fall quickly, we can take only k greatest singular values and corresponding singular vector coordinates and create a *k-reduced singular decomposition* of A.

Definition 2.1 *Let us have $k, 0 < k < r$ and singular value decomposition of A*

$$A = U\Sigma V^T = (U_k U_0) \begin{pmatrix} \Sigma_k & 0 \\ 0 & \Sigma_0 \end{pmatrix} \begin{pmatrix} V_k^T \\ V_0^T \end{pmatrix}$$

*We call $A_k = U_k \Sigma_k V_k^T$ a k-*reduced singular value decomposition *(*rank-k SVD*).*

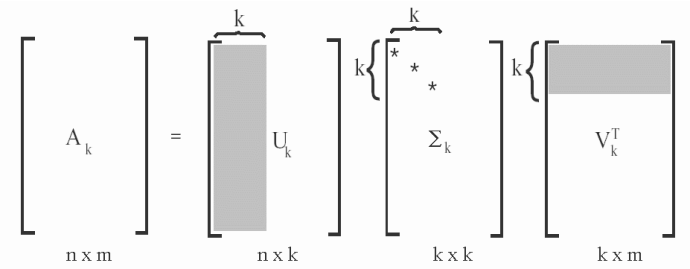


Fig. 1. rank-k Singular Value Decomposition

For an illustration of rank-k SVD see Figure 1, the gray areas determine first k coordinates from singular vectors, which are being used.

Theorem 2.2 (Eckart-Young) *Among all $n \times m$ matrices C of rank at most k A_k is the one, that minimizes $||A_k - A||_F^2 = \sum_{i,j}(A_{i,j} - C_{w,j})^2$.*

Because rank-k SVD is the best rank-k approximation of original matrix A, any other decomposition will increase the sum of squares of matrix $A - A_k$.
SVD rank-k approximation of original matrix A can be understood in several ways. In information retrieval, matrix A represents term-document matrix, and a latent semantics is obtained using rank-k approximation of original matrix A. From another point of view rank-k approximation can be viewed as elimination of noise from data represents in matrix A.
The SVD is computed by a batch $O(nm^2 + n^2m + m^3)$ time algorithm (Golub & Loan, 1996), that is unfeasible for large datasets, but for our case this algorithm is fully adequate.

Note 2.1 *From now on, we will assume rank-k singular value decomposition when speaking about SVD.*

2.2 Tensors and HOSVD

A tensor is a higher order generalization of a vector (first order tensor) and a matrix (second order tensor). Higher order tensors are also called multidimensional matrices or multi-way arrays. The order of a tensor $\mathcal{A} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ is N. Elements of $\mathcal{A}$ are denoted as $a_{i_1 \cdots i_n \cdots i_N}$ where $1 \leq i_n \leq I_n$. In tensor terminology, matrix column vectors are referred to as mode-1 vectors and row vectors as mode-2 vectors. The mode-n vectors of an N-th order tensor $\mathcal{A}$ are the I_n-dimensional vectors obtained from $\mathcal{A}$ by varying the index i_n and keeping the other indices fixed, that is the column vectors of n-mode matrix unfolding $\mathcal{A}_{(n)} \in R^{I_n \times (I_1 I_2 \cdots I_{n-1} I_{n+1} \cdots I_N)}$ of tensor $\mathcal{A}$. See (De Lathauwer et al., 2000) for details on matrix unfoldings of a tensor.
The n-mode product of a tensor $\mathcal{A} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ by a matrix $M \in \mathbb{R}^{J_n \times I_n}$ is an $I_1 \times I_2 \times \cdots \times I_{n-1} \times J_n \times I_{n+1} \times \cdots \times I_N$-tensor of which the entries are given by

$$(\mathcal{A} \times_n M)_{i_1 \cdots i_{n-1} j_n i_{n+1} \cdots i_N} = \sum_{i_n} a_{i_1 \cdots i_{n-1} i_n i_{n+1} \cdots i_N} m_{j_n i_n} \tag{2}$$

Note that the n-mode product of a tensor and a matrix is a generalization of the product of two matrices. It can be expressed in terms of matrix unfolding:

$$B_{(n)} = M A_{(n)} \tag{3}$$

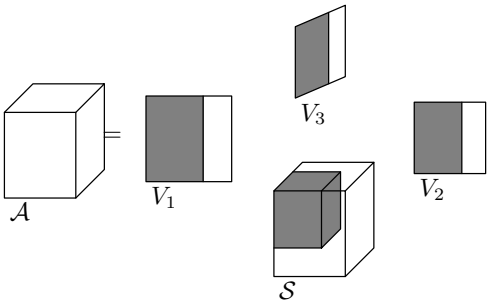


Fig. 2. 3-order HOSVD

where $B_{(n)}$ is the n-mode unfolding of tensor $\mathcal{B} = \mathcal{A} \times_n M$.

In terms of n-mode products, the matrix SVD can be rewritten as $F = S \times_1 V^{(1)} \times_2 V^{(2)}$. By extension, *High order singular value decomposition* (*HOSVD*) is a generalization of matrix SVD: every $I_1 \times I_2 \times \cdots \times I_N$ tensor $\mathcal{A}$ can be written as the n-mode product (De Lathauwer et al., 2000):

$$\mathcal{A} = S \times_1 V^{(1)} \times_2 V^{(2)} \cdots \times_N V^{(N)} \tag{4}$$

HOSVD is illustrated in Figure 2 for $N = 3$. V_n contains the orthonormal vectors (called n-mode singular vectors) spanning the column space of the matrix $A_{(n)}$ (n-mode matrix unfolding of tensor $\mathcal{A}$). $\mathcal{S}$ is called core tensor. Instead of being pseudodiagonal (nonzero elements only occur when the indices satisfy $i_1 = i_2 = \cdots = i_N$), $\mathcal{S}$ has the property of all-orthogonality. That is, two subtensors $\mathcal{S}_{i_n} = \alpha$ and $\mathcal{S}_{i_n} = \beta$ are orthogonal for all possible values of n, α and β subject to $\alpha \neq \beta$. At the same time, the Frobenius-norms $\sigma_i^n = ||\mathcal{S}_{i_n=i}||$ are n-mode singular values of $\mathcal{A}$ and are in decreasing order: $\sigma_1^n \geq \sigma_2^n \geq \cdots \geq \sigma_{I_n}^n \geq 0$ $\mathcal{S}$ is in general a full tensor and governs the interactions among V_n.

3. SVD and HOSVD Approximation of SOM

3.1 BMU Movements

BMU is found for each training vector during learning process. As the SOM learns the structure of training set, the BMU of given training vector usually changes its position within the map. Movement of the BMU at the initial phase of learning process will be probably very rapid, and as the SOM converges to stable configuration the movement of the BMU will be very tight. The learning process could be stopped, when user specific maximal number of moved BMUs is reached.

In this way, the number of moved BMUs can be taken as alternative learning process quality measurement. The number of changes of BMUs' positions between successive iterations was considered as measure in our initial work. But this approach is not very helpful. Some movements of the BMU still remain.

3.2 SVD Approach

The second approach to measurement of BMU movement uses SVD decomposition of SOM. It is supposed, that movement of BMUs among original SOM and its rank-k approximations will be very low, when stable configuration of the SOM is reached.

To verify this hypothesis following experiment was performed:

1. The SOM S is transformed to $rc \times m$ matrix A, where r is the number of rows, c is the number of columns of SOM S, and m is the dimension of input.

$$A = \begin{pmatrix} \mathbf{m}_{1,1}(t) \\ \mathbf{m}_{1,2}(t) \\ \vdots \\ \mathbf{m}_{1,c}(t) \\ \mathbf{m}_{2,1}(t) \\ \vdots \\ \mathbf{m}_{r,c}(t) \end{pmatrix} \tag{5}$$

Note 3.1 *Each* $\mathbf{m}_{i,j}(t)$ *is m dimensional vector.*

2. Rank k approximation A_k of matrix A is computed, $1 \leq k \leq m$.
3. SOM $S(k)$ is created from matrix A_k.
4. The BMU_S for each training vector is computed with original SOM S.
5. The $BMU_S(k)$ is computed with rank k approximation $S(k)$ for each training vector.
6. If BMU_S is different from $BMU_S(k)$ the movement is encountered.

3.3 HOSVD Approach

The SOM forms usually 2D dimensional grid, each neuron is represented as m dimensional vector, where m is the dimension of input, training, vectors. From this point of view, SOM can be understood as 3-order tensor, and 3-dimensional HOSVD, see Figure 2, can be applied onto SOM. There is no need to form matrix by transformation of SOM. 3-order HOSVD can directly decompose SOM. It is expected that HOSVD preserves better relationships among neurons and structure of SOM.
Experiment similar as in SVD approach was performed:

1. The SOM S is transformed to tensor $\mathcal{A} \in R^{r \times c \times m}$, where r is the number of rows, c is the number of columns of SOM S, and m is the dimension of input.
2. $\mathcal{A}_{(k_1,k_2,k_3)}$ approximation of tensor $\mathcal{A}$ was computed. This approximation generalizes rank-k approximation of matrix in SVD.
3. SOM $S(k_1,k_2,k_3)$ is created from tensor $\mathcal{A}_{(k_1,k_2,k_3)}$.
4. The BMU_S for each training vector is computed with original SOM S.
5. The $BMU_S(k_1,k_2,k_3)$ is computed with approximated SOM $S(k_1,k_2,k_3)$ for each training vector.
6. If BMU_S is different from $BMU_S(k_1,k_2,k_3)$ the movement is encountered.

4. Experimental Results

A number of experiments were carried out to prove our hypothesis. One of them is provided in this chapter. Parameters of used SOM S are given in Table 1. The input data comes from experiments done by Kudelka et al. (Kudelka et al., 2006).

Parameter	Value
# of rows	50
# of columns	50
SOM shape	toroid
input dimension	10
# of input vectors	approx 25,000
# of iterations	5000

Table 1. Experimental SOM S parameters

4.1 Results of SVD Approach

The experimental results are summarized in following tables and graphs. Table 2 provides insight to learning process. Each rank-k, $1 \leq k \leq 10$, approximation $S(k)$ of SOM S is computed and number of moved BMUs are stored, moreover a distance of the movement.
In initial phase of learning process, there is no significant difference among $S(k)$ approximations. The number of moved BMUs varies from 99% to 26% in iteration 0. Average distance of movement is also near constant, from 13.6 to 15.4.
There are significant difference among each $S(k)$ approximations after 4000 and 5000 iterations. $S(1)$ approximation has 76% of moved input vectors, but $S(8)$ resp. $S(9)$ has only 9% resp. 2.6% of moved input vectors after 5000 iterations. The average distance is also very small, only 2.4 resp. 3.
Although maximal distance of the movement remains very high in all phases of learning process, the average distance of BMUs movement decreases. Histograms of these distances can provide very useful information about learning. Figure 3(a) demonstrates initial state of SOM S. The distribution of distances is random; there are plenty of short movements and also a lot of long movement of BMUs. The charts 3(d) have different shape. Histogram of distances has strong hyperbolic shape, i.e. very short movements are predominate. Cumulative histogram shows, that movements of length 1 (less than 20%) comprise slightly more than 80% of all movements for $S(8)$ resp. $S(9)$ approximations. These results coincide with the widely known "Pareto's 80/20 law". In other words, SOMs $S(8)$ resp. $S(9)$ are very close to original SOM S, consequently SOM S after 5000 iterations completed contains very small amount of noise (compare $S(9)$ approximation after 500 iterations completed). We can conlude, that well trained SOM is resistant to SVD decomposition.
U-matrices (Ultsch, 1993) of experimental SOM S and its approximations $S(k)$ are given in Figure 4. These figures show, that even $S(1)$ approximation is used, the main features, main structure, of original SOM is preserved. $S(2)$ resp. $S(3)$ approximation add more details to basic structure of $S(1)$ approximation. $S(8)$ resp. $S(9)$ approximations are not easily distinguishable from original SOM.

4.2 Results of HOSVD Approach

On the other hand performance of HOSVD approximation of SOM S was disappointing. U-matrices of approximated SOM are very similar to SVD ones, see Figure 6. Also cumulative histograms 5 of BMUs' movements show very similar characteristics. But when absolute values are compared, there are very high differences, see Table 3 e.g. only 616 BMUs were moved after 5,000 iterations in rank-9 SVD approximation, which is only 2.63 % of all input vectors. But in similar HOSVD approximation more than 9,000 BMUs were moved.

Acknowledgement

This work is supported by Grant of Grant Agency of Czech Republic No. 205/09/1079.

5. Conclusion

Alternative approach of quality measurement of the SOM learning process was presented. This approach uses SVD decomposition of the SOM, and number of moved BMUs are counted among each consecutive rank-*k* approximations of original SOM. Our experiments show some properties of SVD decomposition of SOM. That is new and good.
On the other hand HOSVD cause some problems, because number of moved BMUs is very high. It seems that SOM is not true three dimensional object – the third dimension consisting of neurons is not relevant. We should check if this behavior is feature or bug of this method or if it is a matter of coincidence. This is new and quite disappointing.
Usage of Non-negative Matrix Factorization (Golub & Loan, 1996) is our goal in future research. The second goal is application of presented method on the other type of neural network, such as Growing Neural Gas (Fritzke, 1994).

6. References

Bauer, H. U., Herrmann, M. & Villmann, T. (1999). Neural maps and topographic vector quantization, *Neural Networks* **12**(4): 659–676.

Berry, M. & Browne, M. (1999). *Understanding Search Engines, Mathematical Modeling and Text Retrieval*, Siam.

Berry, M., Dumais, S. & Letsche, T. (1995). Computation Methods for Intelligent Information Access, *Proceedings of the 1995 ACM/IEEE Supercomputing Conference*.

Cottrell, M., Fort, J. C. & Pages, G. (1998). Theoretical aspects of the SOM algorithm, *Neurocomputing* **21**(1): 119–138.

de Bodt, E. & Cottrell, M. (2000). Bootstrapping self-organising maps to assess the statistical significance of local proximity, *8th European Symposium on Artificial Neural Networks. ESANN"2000. Proceedings. D-Facto, Brussels, Belgium*, pp. 245–54.

De Lathauwer, L., Moor, B. D. & Vandewalle, J. (2000). A multilinear singular value decomposition, *SIAM Journal on Matrix Analysis and Applications* **21**(4): 1253–1278.
URL: *citeseer.ist.psu.edu/delathauwer00multilinear.html*

Dvorský, J. (2007). Self-organizing maps and svd, *DEXA Workshops*, IEEE Computer Society, pp. 143–147.

Flanagan, J. A. (1994). *Self-Organizing Neural Networks*, PhD thesis, École Polytechnique Fédérale de Lausanne, Lausanne.

Flanagan, J. A. (1996). Self-organization in Kohonen's SOM, *Neural Networks* **9**: 1185–1197.

Fritzke, B. (1994). A growing neural gas network learns topologies, *in* G. Tesauro, D. S. Touretzky & T. K. Leen (eds), *NIPS*, MIT Press, pp. 625–632.

Germen, E. (2005). Improving the resultant quality of kohonen's self organizing map using stiffness factor, *in* L. Wang, K. Chen & Y.-S. Ong (eds), *ICNC (1)*, Vol. 3610 of *Lecture Notes in Computer Science*, Springer, pp. 353–357.

Golub, G. H. & Loan, C. F. V. (1996). *Matrix Computations*, Johns Hopkins University Press.

Kohonen, T. (2001). *Self-Organizing Maps*, 3rd edn, Springer-Verlag New York, LLC.

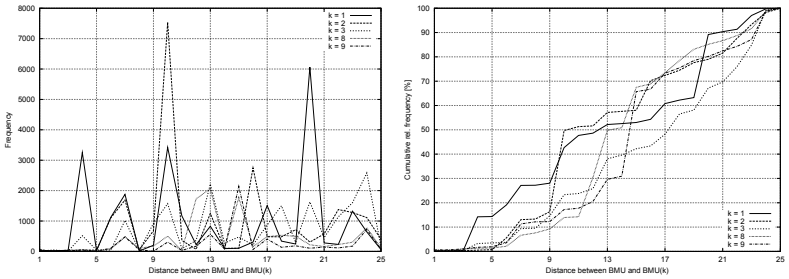
Kudelka, M., Snasel, V., Lehecka, O. & El-Qawasmeh, E. (2006). Semantic analysis of web pages using web patterns, *2006 IEEE/WIC/ACM International Conference on Web Intelligence* **0**: 329–333.

Larsen, R. M. (1998). Lanczos bidiagonalization with partial reorthogonalization, *Technical report*, University of Aarhus.

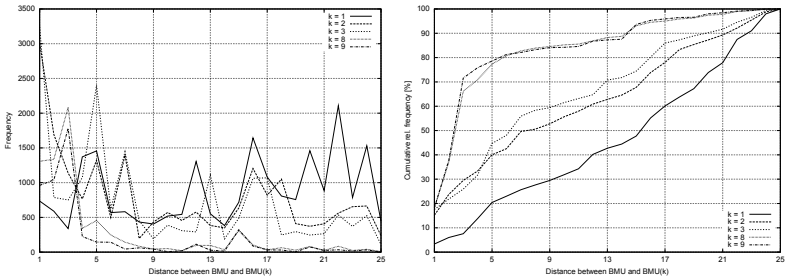
Mulier, F. M. & Cherkassky, V. S. (1995). Statistical analysis of self-organization, *Neural Networks* **8**(5): 717–727.

Rynkiewicz, J. (2006). Self-organizing map algorithm and distortion measure, *Neural Networks* **19**(6-7): 830–837.

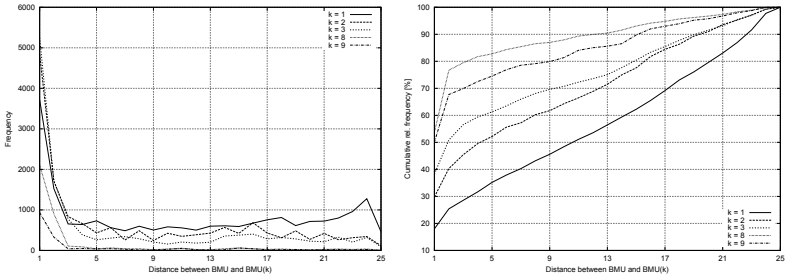
Ultsch, A. (1993). Self-organizing neural networks for visualization and classification, *in* O. Opitz, B. Lausen & R. Klar (eds), *Information and Classification*, Springer, London, UK, pp. 307–313.



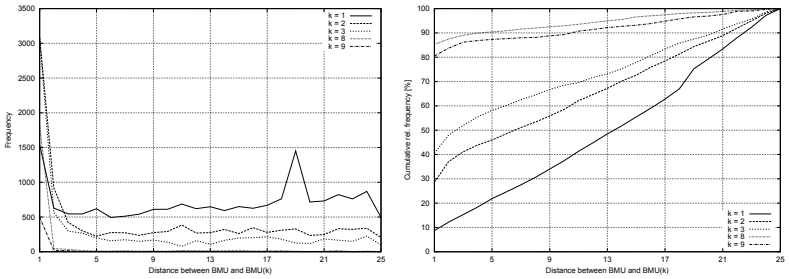
(a) Initial SOM with random weights



(b) SOM after 500 iterations

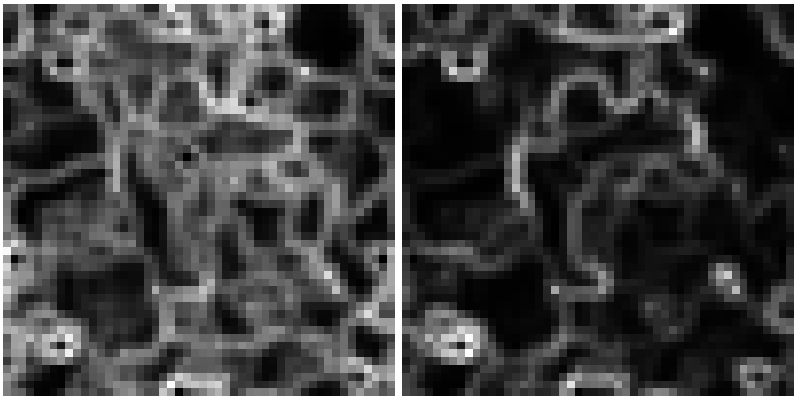


(c) SOM after 4000 iterations

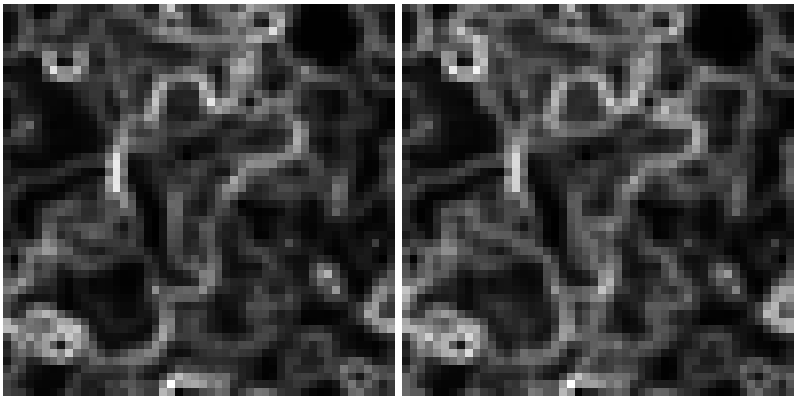


(d) SOM after 5000 iterations (final state)

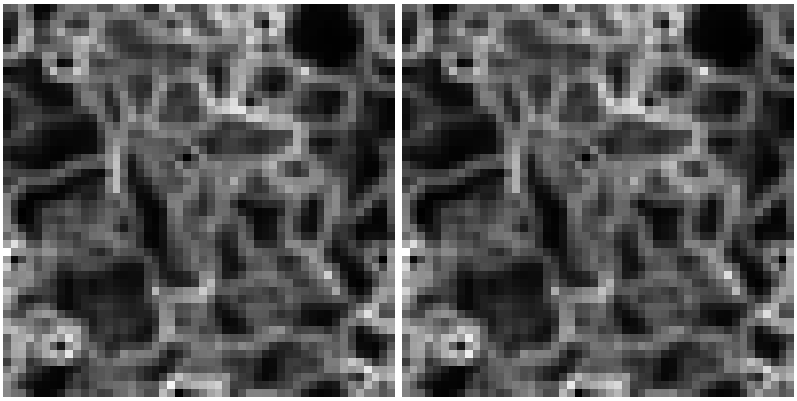
Fig. 3. Frequency and cumulative frequency histograms of distances between BMU_S and $BMU_S(k)$ for SOM S



(a) Original SOM (b) $k = 1$ approximation

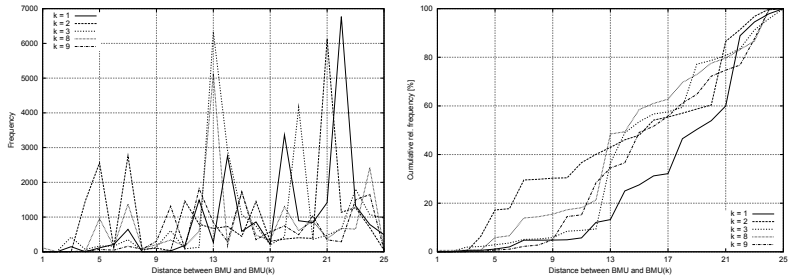


(c) $k = 2$ approximation (d) $k = 3$ approximation

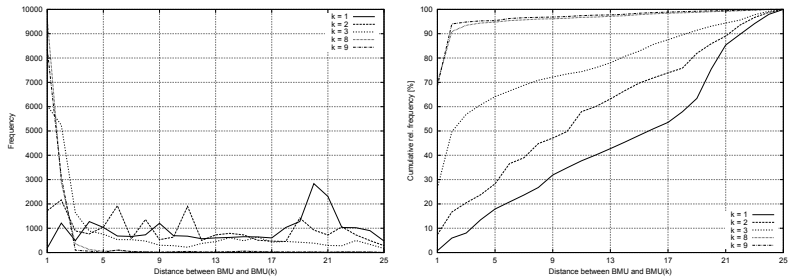


(e) $k = 8$ approximation (f) $k = 9$ approximation

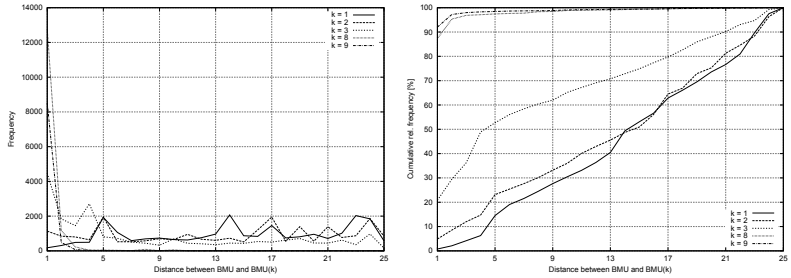
Fig. 4. SVD rank-k approximations of final SOM S (after 5000 iterations)



(a) Initial SOM with random weights

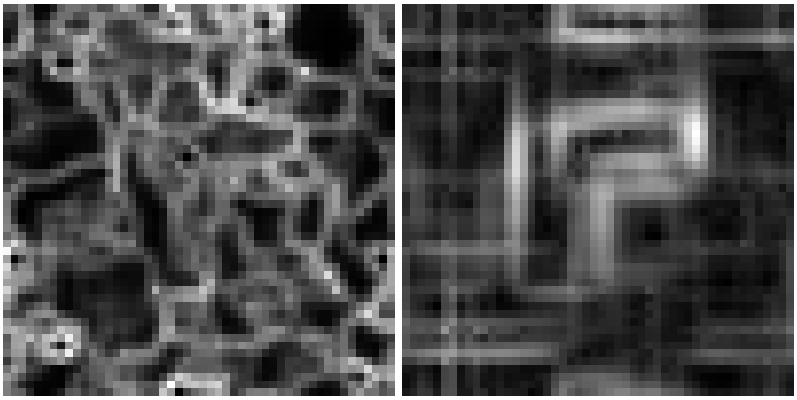


(b) SOM after 4000 iterations

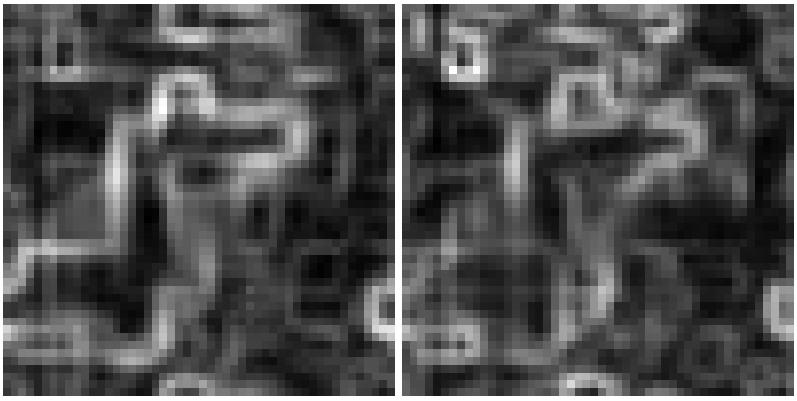


(c) SOM after 5000 iterations (final state)

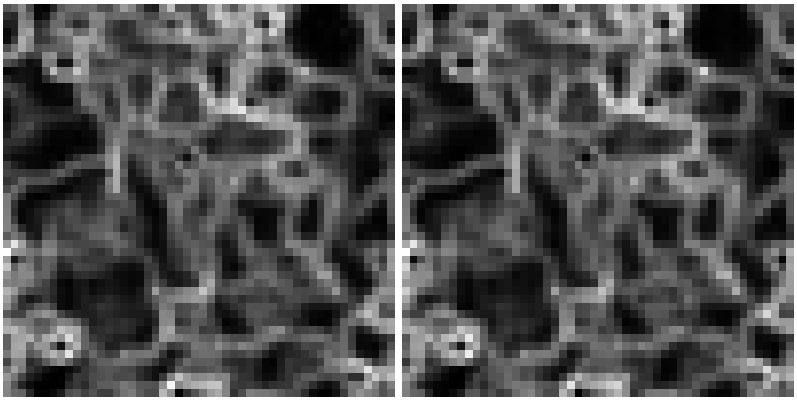
Fig. 5. Frequency and cumulative frequency histograms of distances between BMU_S and $BMU_S(k_1,k_2,k_3)$ for SOM S



(a) Original SOM (b) $k_1 = 5, k_2 = 5, k_3 = 1$ approximation



(c) $k_1 = 10, k_2 = 10, k_3 = 2$ approximation (d) $k_1 = 15, k_2 = 15, k_3 = 3$ approximation



(e) $k_1 = 40, k_2 = 40, k_3 = 8$ approximation (f) $k_1 = 45, k_2 = 44, k_3 = 9$ approximation

Fig. 6. $S(k_1, k_2, k_3)$ HOSVD approximations of final SOM S (after 5000 iterations)

	Number of singular values k									
	1	2	3	4	5	6	7	8	9	10
	0 completed iterations									
MSE	0.784	0.74	0.712	0.64	0.614	0.57	0.536	0.537	0.522	0.517
# of moved BMUs	23,381	22,570	18,053	15,897	14,963	12,326	11,031	10,654	6,162	0
Moved BMUs [%]	99.825	96.362	77.077	67.872	63.884	52.626	47.097	45.487	26.309	0
Max. distance	25	25	25	25	25	25	25	25	25	
Average distance	13.6	13.9	16.6	17.7	17.2	16.3	17	14.9	15.4	
	500 completed iterations									
MSE	0.603	0.456	0.366	0.303	0.244	0.209	0.172	0.132	0.106	0.07
# of moved BMUs	22,000	19,805	18,468	16,496	13,823	11,681	9,788	7,147	5,283	4
Moved BMUs [%]	93.929	84.557	78.849	70.43	59.017	49.872	41.79	30.514	22.556	0.017
Max. distance	25	25	25	25	25	25	25	25	24	1
Average distance	14.3	10	9.2	6.6	6.4	5.9	5.5	5	4.8	1
	4000 completed iterations									
MSE	0.595	0.452	0.356	0.284	0.236	0.184	0.143	0.092	0.04	0.015
# of moved BMUs	20,643	16,252	13,698	11,964	9,353	8,185	4,907	3,925	1,865	0
Moved BMUs [%]	88.135	69.388	58.483	51.08	39.933	34.946	20.95	16.758	7.963	0
Max. distance	25	25	25	25	25	25	25	25	24	
Average distance	11.4	8	6.9	5.8	4.8	3.8	4.2	3.7	4.7	
	5000 completed iterations									
MSE	0.595	0.452	0.355	0.284	0.229	0.18	0.119	0.083	0.038	0.005
# of moved BMUs	1,7748	10,760	7,555	6,868	5,058	3,781	2,564	2,125	616	1
Moved BMUs [%]	75.775	45.94	32.256	29.323	21.595	16.143	10.947	9.073	2.63	0.004
Max. distance	25	25	25	25	25	25	25	25	25	1
Average distance	13.2	9.1	7.5	5.5	5	4	3.7	2.4	3	1

Table 2. Parameters of SVD rank-k approximations of given SOM

	Approximation parameters (k_1, k_2, k_3)								
	(5, 5, 1)	(10, 10, 2)	(15, 15, 3)	(20, 20, 4)	(25, 25, 5)	(30, 30, 6)	(35, 35, 7)	(40, 40, 8)	(45, 45, 9)
	0 completed iterations								
MSE	1.485	1.218	1.136	0.979	0.887	0.754	0.696	0.665	0.602
# of moved BMUs	23,422	23,421	23,407	23,104	23,299	23,109	21,829	18,876	13,799
Moved BMUs [%]	100	99.996	99.936	98.642	99.475	98.664	93.199	80.591	58.915
Max. distance	25	25	25	25	25	25	25	25	25
Average distance	18.4	14.6	16.3	16.7	15	15.9	15.9	15.4	16.7
	4000 completed iterations								
MSE	0.595	0.452	0.356	0.285	0.237	0.184	0.143	0.092	0.04
# of moved BMUs	23,404	23,304	22,722	21,671	20,141	18,890	17,265	13,761	12,218
Moved BMUs [%]	99.923	99.496	97.011	92.524	85.992	80.651	73.713	58.752	52.165
Max. distance	25	25	25	25	25	25	25	25	25
Average distance	14.3	11	6.6	6.5	5	2.5	2.3	2	1.8
	5000 completed iterations								
MSE	0.595	0.453	0.356	0.285	0.23	0.182	0.12	0.084	0.039
# of moved BMUs	23,398	23,220	21,419	18,217	19,305	18,132	15,638	14,639	9,252
Moved BMUs [%]	99.898	99.138	91.448	77.777	82.423	77.414	66.766	62.501	39.501
Max. distance	25	25	25	25	25	25	25	25	25
Average distance	14.6	13.7	8.6	6	3.4	2.2	1.8	1.4	1.3

Table 3. Parameters of $S(k_1, k_2, k_3)$ HOSVD approximations of given SOM S

PartSOM: A Framework for Distributed Data Clustering Using SOM and K-Means

Flavius L. Gorgônio and José Alfredo F. Costa
Federal University of Rio Grande do Norte
Brazil

1. Introduction

Cluster analysis can be defined as the process of partition data into a certain number of clusters (or groups) of similar objects, where each group consists of similar objects amongst themselves (internal homogeneity) and different from the objects of the other groups (external heterogeneity), i.e., patterns in the same cluster should be similar to each other, while patterns in different clusters should not (Xu & Wunsch II, 2005). Typical clustering applications include pattern recognition, data mining, data compression, market segmentation and dimensionality reduction, among others.

More formally, given a set of N input patterns: $X = \{x_1, \ldots, x_N\}$, where each $x_i = (x_{i1}, \ldots, x_{ip})^{\mathrm{T}} \in \Re^p$ represents a p-dimensional vector and each measure x_{ij} represents a attribute (or variable) from dataset, a clustering process attempts to seek a K partition of X, denoted by $C = \{C_1, \ldots, C_K\}$, $(K \leq N)$ such that:

a) $C_i \neq \phi, i = 1 \ldots K;$

b) $\bigcup_{i=1}^{K} C_i = X;$

c) $C_i \cap C_j = \phi, \ \ i, j = 1 \ldots K; i \neq j.$

Traditional clustering algorithms operate over a single dataset. In most cases, before applying traditional algorithms on distributed databases, all distributed data must be centered on a main site. However, in some applications, e.g. medical and business, privacy-preserving and security policies disallow data combination, because confidential information can be reconstructed (Silva & Klusch, 2006).

In very large databases, the integration of several datasets into a single location is discouraged (Forman & Zhang, 2000). If an organization has very large distributed databases and needs to gather all the data to apply clustering algorithms, processing can demand excessive data transfers, which can be slow and expensive.

Moreover, any change that occurs in the distributed data, as for example, the inclusion of new information or alterations of existing data, will have to be updated into the central database. This demands a complex process of information updating, which causes a data transfer overload within the system.

For that reason, several algorithms have appear aimed at clustering dispersed data into several locations and summarizing results in a central unit, ensuring data security and confidentiality. This approach is known as distributed data clustering (DDC).
Clustering algorithms can be based on a wide variety of theories and techniques, including graph theory, combinatorial search techniques, fuzzy set theory, artificial neural networks, and kernels techniques (Xu & Wunsch II, 2005). Artificial neural networks are an important computational tool, with strong inspiration neurobiological and widely used in the solution of complex problems, which cannot be handled with traditional algorithmic solutions (Haykin, 1999). Applications for neural networks include pattern recognition, signal analysis and processing, analysis tasks, diagnosis and prognostic, data classification and clustering.
Competitive neural networks provide a family of algorithms used for data representation, visualization and clustering. Among the unsupervised neural network models, the self-organizing map (SOM) plays a major role. SOM features include information compression while trying to preserve the topological and metric relationship of the primary data space (Kohonen, 2001). The SOM network defines, via unsupervised learning, a mapping of a continuous p-dimensional space to a set of model vectors, or neurons, usually arranged as a 2-D array.
This work proposes a novel strategy for cluster analysis in distributed databases using a recently proposed architecture, named *partSOM*, and typical clustering algorithms, such as SOM and K-Means. In this approach, the clustering algorithm is applied separately in each distributed dataset, relative to database vertical partitions, to obtain a representative subset of each local dataset. In the sequence, these representative subsets are sent to a central site, which performs a fusion of the partial results. Next, a clustering algorithm is applied again to obtain a final result.
The main contribution of this paper is to show that, in situations where the volume of data is very large or when data privacy and security requirements impede consolidation at a single location, the results obtained with the application of this strategy justify its use.
The remainder of the chapter is organized as follows: section 2 presents a brief bibliographical review about distributed data clustering algorithms and section 3 describes the main aspects of the SOM and K-Means algorithm. Section 4 presents the proposed strategy, detailing its operation and the advantages obtained with its use. Section 5 describes the methodology used in the experiments and section 6 present the results of the application of the proposed strategy for some datasets, comparing them with the results obtained with the traditional approaches. Finally, section 7 presents conclusions and the direction of future algorithm research.

2. A Review on Distributed Data Clustering

Cluster analysis algorithms groups data based on the similarities between objects or patterns. The complexity of the cluster analysis process increases with data cardinality (N, the number of objects in a database) and dimensionality (p, the number of attributes). Clustering methods range from those that are largely heuristic to more formal procedures based on statistical models. The most frequently used methods are hierarchical (or heuristic) and partitioning (or iterative) methods. Several algorithms have been developed based on different strategies, including hierarchical clustering, vector quantization, graph theory,

fuzzy logic, artificial neural networks, combinatory search, and others. Xu & Wunsch II (2005) and Berkhin (2006) present two recent surveys of clustering algorithms.

Hierarchical methods have been dominant in the clustering literature and proceed by stages, producing a sequence of partitions, each corresponding to a different number of clusters (Costa & Andrade Netto, 2001). Partitioning methods produce one partition with *K* groups, usually by minimizing some objective criterion. The most common method is the K-Means, which uses heuristics for reducing the within-group sum of squares. Each method has its advantages and drawbacks. Most partitional methods, for example, require a priori choice of the number of clusters and may be sensitive to initialization. Furthermore, many validation criteria are currently available, but if left to the user's choice, may result in disagreement due to the possible differences of the geometric structure imposed on the data space from clustering algorithm and the validation index.

Searching for clusters in high-dimensional databases is a non trivial task. Some common algorithms, such as traditional agglomerative hierarchical methods, are unsuitable for large datasets. Moreover, the increase in the number of attributes of each entrance not only has a negative influence on the processing time of the algorithm, but also hinders cluster identification. An alternative approach is to divide the database into partitions and to perform data clustering of each one separately.

Some current applications have databases that are large enough that cannot be maintained integrally in the main memory, even on very powerful computers. Kantardzic (2003) describe three approaches to solving this problem:

a. The data are stored in secondary memory and data subsets are clustered separately, obtaining the results in a subsequent stage that contains the whole group;
b. An algorithm of incremental grouping is used. Each element is individually stored in the main memory and associated to one of the existing groups or allocated to a new group;
c. A parallel implementation is used and several algorithms work simultaneously on the stored data.

Two approaches are frequently used to partition dataset: the first, and more common, is to divide the database horizontally, creating homogeneous subsets of the data, so that each algorithm operates on the same attributes, although treating of different registers. Another approach is to divide the database vertically, creating heterogeneous data subsets. In this case, each algorithm operates on the same registers, but handles on different sets of attributes.

There have been recent published studies about distributed data clustering. Forman & Zhang (2000) present a method that parallels several algorithms to obtain greater efficiency in the data mining process of distributed databases. These authors reinforce the need for reducing communication overload among the bases, to reduce processing time and to minimize the need for powerful machines with extended storage capacities.

Other factors that motivate the existence of distributed databases are related to security and privacy (Lam et al., 2004). Privacy-preserving clustering focuses its efforts on algorithms that ensure data privacy and security, as for example, in medical or business databases. Several organizations maintain geographically distributed databases as a form of increasing the security of their information. Thus, if security fails, the intruder can access only part of the information.

İnan et al. (2007) present a method for clustering horizontally partitioned databases, based on constructing the dissimilarity matrix of objects from different sites, which can be used for privacy preserving clustering. Vaidya & Clifton (2003) presents an approach for vertically partitioned databases using a distributed K-Means algorithm. On the other hand, Jagannathan et al. (2006) present a K-means variant algorithm for clustering horizontally partitioned databases. Oliveira & Zaïane (2004) propose a spatial data transformation method for protecting attributes values when sharing data for clustering, called RBT, which is independent of any clustering algorithm.

In databases with a large number of attributes, another approach is sometimes used to perform the analysis considering only an attribute subset, instead of considering all of them. An obvious difficulty with this approach is to identify which attributes are the most important in the cluster identification process. Some studies have used statistical methods such as Principal Components Analysis (PCA) and Factor Analysis to deal with this problem (Friedman & Meulman, 2004; Damian et al., 2007).

Kargupta et al. (2001) describe a PCA-based technique, called Collective Principal Component Analysis (CPCA), to cluster high-dimensional heterogeneous distributed databases. The authors focus on reducing data transfer rates between distributed sites.

He et al. (2005) analyze the influence of data types on the clustering process and present a strategy for dividing the group of attributes into two subsets, one with only the numerical attributes and the other with only the categorical ones. Authors then propose to cluster separately each of these subsets, using appropriate algorithms for each type. In the end, the results of each cluster are combined into a new database, which is once again submitted to a clustering algorithm for categorical data.

*Part*SOM is a simple and efficient architecture to cluster distributed datasets, based on multiples self-organizing maps disposed in a parallel arrangement (Gorgônio, 2009). This architecture applies SOM algorithm separately in each distributed dataset, corresponding to vertical partitions of data, to obtain a representative subset of each local dataset. In the sequence, these representative subsets are sent to a central site, which performs a fusion of the results and applies the SOM algorithm again to obtain the final result (Gorgônio and Costa 2008a; Gorgônio and Costa, 2008b).

There are two main advantages of the *part*SOM architecture over traditional approaches. At first, since only a reduced data volume is transferred between local and central units, the architecture is efficient in situations where the data volume is very large. Second, since only representative pattern are sent to central unit, the architecture is particularly interesting when data privacy and security policies disable data consolidation into a single location.

This work extends this approach, showing that strategy is efficient not only with SOM, but also using others clustering algorithms, such as K-Means or a combination of both in the same process. A series of experiments we carry out with the objective to determine the efficacy of proposed strategy, using a set of public domain databases and experimental results are compared with traditional SOM and K-Means approaches.

3. Clustering Algorithms

3.1 K-Means

K-Means is the most commonly-used clustering algorithm, particularly for its simplicity and ease of implementation. The algorithm objective is to group a set of *N* item in *K* groups, based on some similarity measure, normally the Euclidean distance, given for:

$$\left\| x_i - x_j \right\| = \sqrt{\sum_{f=1}^{p} \left(x_{if} - x_{jf} \right)^2} \tag{1}$$

where x_i and x_j represent two elements from dataset and x_{if} represents the *f*-th attribute from *i*-th element.

The algorithm starts choosing a random set of *K* centroids, where *K* value is previously defined by the user. Next, the algorithm calculates a distance matrix between each one of the dataset elements and the centroids.

In the following step, each element is associated with its nearest centroid. The value of each centroid then is recomputed, based on mean of the values of the elements that belongs to this centroid. A new distances matrix is calculated and the process repeats until the convergence. The algorithm finishes when each element is associated to the cluster represented for its centroid.

3.2 Self-Organizing Maps

The self-organizing feature map (SOM) has been widely studied as a software tool for visualizing high-dimensional data. Important features include information compression while preserving the topological and metric relationship of the primary data items (Kohonen, 2001). The SOM is composed of two layers of neurons: an input layer and an output (or competitive) layer. The output layer neurons are connected to adjacent neurons by a neighboring relation, defining the topology of the map.

Training is accomplished by presenting one input pattern x at a time in a random sequence and comparing it, in parallel, with all the reference vectors. The best match unit (BMU), which can be calculated using the Euclidean metric, represents the weight vector with the greatest similarity to that input pattern. Denoting the winning neuron by c, the BMU can be formally defined as the neuron for which

$$\left\| x - x_c \right\| = \min_i \left\{ \left\| x - x_i \right\| \right\} \tag{2}$$

where $\|\cdot\|$ is the measure of distance.

The input is thus mapped to this location. The weight vectors of BMU as well as the neighboring nodes are moved closer to the input data vector. The magnitude of the attraction is governed by the learning rate. The SOM update rule for the weight vector of the unit *i* is

$$x(t+1) = x_i(t) + h_{ci}(t) \cdot \left[x(t) - x_i(t) \right] \tag{3}$$

where t denotes time, $x(t)$ is the input vector randomly drawn from the input data set at time t and $h_{ci}(t)$ is the neighborhood kernel around the winning unit c at time t. This last term is a non-increasing function of time and of the distance of unit i from BMU and usually composed of two components: the learning rate function $\alpha(t)$ and the neighborhood function $h(d,t)$:

$$h_{ci}(t) = \alpha(t) \cdot h\left(\left\| r_c - r_i \right\|, t\right) \tag{4}$$

where r_i denotes the location of unit i on the map grid.

As learning proceeds and new input vectors are given to the map, the learning rate $\alpha(t)$ gradually decreases to zero, according to the specified learning rate function type. Along with learning rate, the neighborhood radius decreases as well.

Despite the SOM is an excellent tool in clustering tasks, almost always, another complementary approaches may be needed to increase the obtained results, particularly when there are many clusters or cluster borders are not well defined. A number of methods for visualizing data relations in a trained SOM have been proposed, such as multiple views of component planes, mesh visualization using projections and 2D and 3D surface plots of distance matrices.

The U-matrix method (Ultsch, 1993) enables visualization of the topological relations of the neurons in an organized SOM. A gradient image (2D) or a surface plot is generated by computing the distances between adjacent neurons. High values in the U-matrix encode dissimilarities between neurons and correspond to cluster borders.

A host of strategies for cluster detection using the U-matrix were proposed in the literature (Costa & Andrade Netto, 2001a; Costa & Andrade Netto, 2003). Three main algorithms were presented: mathematical morphology derived map segmentation (Costa & Andrade Netto, 2001b); a graph partitioning approach (Costa & Andrade Netto, 2003) and contiguity-constrained hierarchical clustering approaches (Gonçalves et al., 2008; Murtagh, 1995). Both algorithms were developed for automatic partitioning and labeling of a trained SOM network.

The first approach uses image processing algorithms such as the watershed transform, which are used to obtain connected regions of neurons representing similar stimuli classes. The second approach uses rules to partition the map by analyzing inconsistent neighboring relations between neurons. Each resulting neuron cluster is a sub-graph that defines complex and non-parametric geometries in the input space, which approximately describes the shape of the clusters. Regarding the last approach, Gonçalves et al. (2008) present improvements of contiguity-constrained hierarchical clustering approaches using validation indexes.

Some works apply clustering algorithms over the U-matrix to segment the map in well defined regions. Vesanto & Alhoniemi (2000) propose strategies for cluster detection using different approaches to group similar neurons in the SOM, including the use of hierarchical agglomerative clustering and partitive clustering, using the K-Means algorithm.

4. The Proposed Strategy

Distributed clustering algorithms usually work in two stages. Initially, the data are analyzed locally in each unit that is part of the distributed database. In a second stage, a central instance gathers partial results and combines them into an overall result.

This section presents a strategy for clustering similar objects located in distributed databases using traditional clustering algorithms, such as self-organizing maps and K-Means. An imposed restriction is that proposed strategy requires a prototype-based algorithm to perform the dataset partition into subsets and select a set of one or more representative elements to each cluster. This representative set is normally named prototype (or codebook) and will be used in a process similar to vectorial quantization in order to select a sample of each remote unit.

The entire process is also divided into two stages:

1. The clustering algorithm is applied locally in each one of the distributed bases, in order to elect a representative subset from input data, which is send to central unit;
2. The clustering algorithm is applied again, this time to the representatives of each one of the distributed bases, that are unified in a central unit;

The proposed strategy, consisting of five steps, is presented as follows:

Proposed Strategy

1. Each local site applies any clustering algorithm to its data, obtaining a reference vector (codebook)
2. Each local site codes its data, using the codebook, to obtain a vector of data indexes that will represent the original data
3. Each local site send indexes and reference vectors to the central site
4. The central site remounts the dataset based on each index and reference vectors received

Step 1 applies anyone traditional clustering algorithm in each local dataset, relative to vertical partitions from the database. Thus, the algorithm is applied to an attribute subset in each of the remote units, obtaining a reference vector from each data subset. This reference vector, known as the codebook, contains a representative subset of each local unit.

Step 2 codes the input data of each local unit using the codebook obtained in the previous stage. Each input is presented to the obtained codebook and the index corresponding to the closest vector present in the codebook is stored in an index vector. So, a data index is created based on representative objects instead of original objects. Despite the difference from the original dataset, representative objects in the index vector are very similar to the original data.

In step 3, each remote unit sends its index and reference vector to the central unit, which is responsible for unifying all partial results. An additional advantage of the proposed algorithm is that the amount of transferred data is considerately reduced, since index vectors have only one column (containing an integer value) and the codebook is usually mush less than the original data. So, reducing data transfer and communication overload are considered by the proposed algorithm.

Step 4 is responsible for receiving the index vector and the codebook from each local unit and combining partial results to remount a database based on received data. To remount each dataset, index vector indexes are substituted by the equivalent value in the codebook. Datasets are combined juxtaposing partial datasets; however, it is important to ensure that objects are in the same order as that of the original datasets. Note that the new database is slightly different from the original data, but data topology is maintained, similar to procedure used for vectorial quantization.

In step 5, any clustering algorithm is again applied over the complete database obtained in step 4. The expectation is that the results obtained in that stage can be generalized as being equivalent to the clustering process of the entire database. The data obtained after the step 4 and that will serve as input in stage 5 correspond to values close to the original, because vectors correspondents in codebook are representatives of input dataset.

An overview of the complete architecture is showed in Fig. 1, considering the SOM algorithm. Similarly, K-Means or other prototype-based clustering algorithms can be used in proposed strategy.

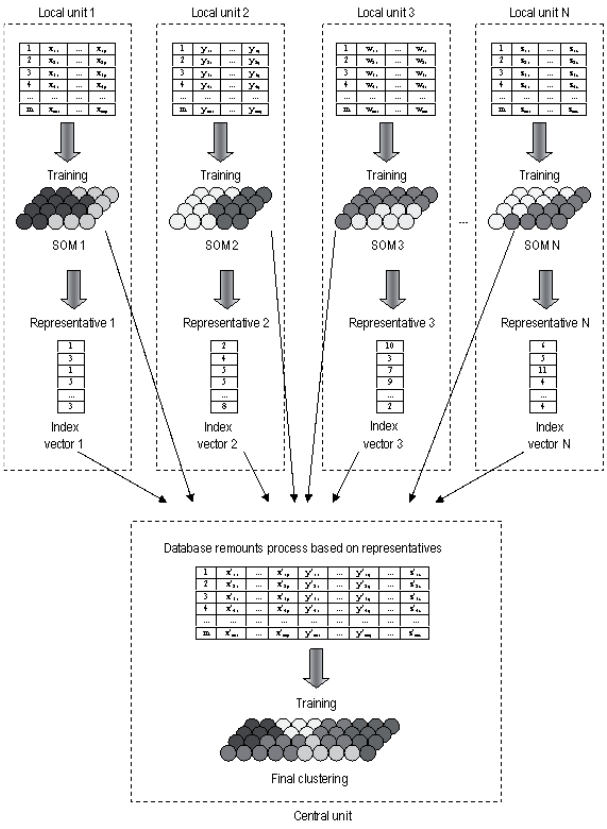


Fig. 1. Complete architecture of proposed strategy using the SOM algorithm

5. Methodology

In order to verify the precision of the proposed strategy, results with this approach are compared with those of traditional clustering using traditional SOM and/or K-Means algorithms. Experiments were carried out using the SOM Toolbox 2.0 package, a SOM implementation for Matlab, available at http://www.cis.hut.fi/projects/somtoolbox/ (Vesanto, 2000).

As a form of validating the proposed strategy, several comparative criteria were used, including the individual counting of errors obtained in the application of the algorithm over well known datasets, the use of quality measures present in SOM toolbox and visual comparison of the trained maps and U-Matrix between the traditional SOM algorithm and the proposed approach.

Several quality measures have been proposed to evaluate and compare clustering algorithms. The SOM Toolbox includes two:

1. Data representation accuracy, measured using average quantization error between data vectors and their BMUs on the map, and
2. Dataset topology representation accuracy, measured using topographic error, which is the percentage of data vectors for which the first and second BMUs are not adjacent units.

The databases used were obtained from the UCI data repository (Asuncion & Newman, 2007). Additional information about number of instances, number of attributes and clusters contained in each dataset are presented in Table 1.

Dataset	Instances	Attributes	Attribute Type	Clusters
Iris	150	4	Numerical	3
Wine	178	13	Numerical	3
Breast Cancer	699	10	Numerical	2
Mushroom	8124	22	Categorical	2

Table 1. Characteristics of datasets used in experiments

6. Experiments

6.1 Iris Dataset

The well-known Iris dataset was used in a variety of studies on pattern recognition. It presents 150 instances containing width and length measures of the sepals and petals of three species of the flower Iris: 'Setosa', 'Versicolor' and 'Virginical'. With 4 attributes and 3 classes, each containing 50 objects, the aim is to cluster similar species based on their measurements. One class is linearly separable 'Setosa' whereas classes 'Versicolor' and 'Virginica' have a degree of mixture.

In the first experiment, Iris dataset was analyzed initially using the traditional SOM algorithm, in a plan hexagonal lattice map with 11 x 6 neurons. Dataset was then vertically partitioned into two subsets, each containing two attributes. The SOM algorithm was applied to two subsets separately and, finally, on the representatives of each one. In this phase, we used two maps, with 9 x 7 and 21 x 3 neurons in the local units and a final map with 11 x 6 neurons, both plan and hexagonal lattice. Maps size was automatic defined by

SOM Toolbox, based on data distribution in input space (Vesanto, 2000). In all experiments maps were randomly initialized and batch SOM algorithm was used.
Two training phases were performed: rough and fine tuning training phases. In rough phase of the traditional SOM approach, neighborhood radius was defined as $\sigma_{initial} = 2$ and $\sigma_{final} = 1$ and *trainlen* was defined to 5 epochs. In fine tuning phase, $\sigma = 1$ and *trainlen* was defined to 18 epochs.
In proposed strategy approach, was used over again a randomly initialized map and batch training method. In rough phase, neighborhood radius was defined as $\sigma_{initial} = 2$ in the first map and as $\sigma_{initial} = 3$ in the second. Final map used same values of the traditional SOM experiment (11 x 6 neurons). Final neighborhood radius was $\sigma = 1$ to all three maps. The *trainlen* parameter was defined as 5 epochs in rough phase in the three maps and as 17 epochs in fine tuning phase for two first maps (encoding phase) and as 18 epochs in the final map.
Two criteria were used to compare the results. The first consists of labeling each of the neurons on the map with information about the class number it belongs to, in agreement with the number of input data instances that it represents. For this to occur, each input data instance had to have a label to identify its class. Note that privileged information was not used during the training phase, only during the algorithm accuracy verification.
In the first approach, traditional SOM was 96% accurate, which corresponds to 6 missing classifications out of 150 instances. In the alternative approach, the accuracy of the proposed strategy increased to 96.7%, which corresponds to 5 missing classifications in the analyzed group.
The second criterion is more subjective, because it is based on a visual comparison. As previously described, the U-matrix enables visualization of clusters (similarity and dissimilarity regions) detected by SOM algorithm. Fig. 2 shows the self-organizing maps obtained with traditional SOM and the proposed strategy using the Iris dataset. Maps were manually colored to identify the class that the neuron belongs to.

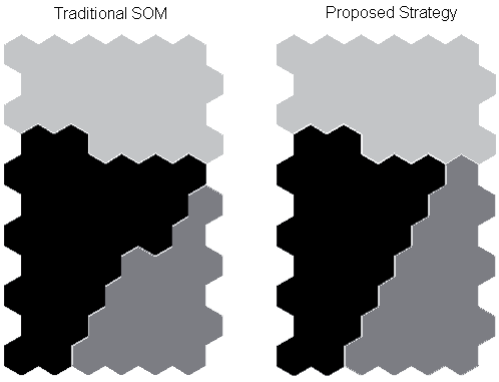


Fig. 2. Comparison between segmented maps to traditional SOM algorithm (left) and proposed strategy (right), using Iris dataset.

In the second experiment, Iris dataset was analyzed using combined SOM and K-Means algorithms. As in previous experiment, the dataset was vertically partitioned into two

subsets, each containing two attributes. The SOM algorithm was applied to two subsets separately and K-Means was applied on the representatives of each one. Maps size and lattice were identical to those defined in the previous experiment. In the central unit, centroids number was defined as $k = 3$, identical to classes number.
Table 2 summarize average values obtained in the first experiment and present other clustering quality measures like QE (average quantization error) and TE (topographic error).

Algorithm	QE	TE	Errors	Hits (%)
Traditional SOM approach	0.3927	0.0133	6	96.0%
Proposed Strategy with SOM + SOM	0.2934	0.0067	5	96.7%

Table 2. Comparatives missing cluster values betweens traditional SOM and the proposed strategy (Iris Dataset)

Table 3 summarize average values obtained in the second experiment, using a combined approach with SOM and K-Means algorithms.

Algorithm	Errors	Hits (%)
Traditional K-Means approach	27	82.0%
Proposed Strategy with SOM + K-Means	25	83.3%

Table 3. Comparatives missing cluster values betweens traditional K-Means and the proposed strategy (Iris Dataset)

6.2 Wine Dataset

The Wine dataset has 178 instances and 13 attributes, which correspond to the results of chemical analyses performed with three types of wines that are produced in the same region of Italy, but from different cultivations. Attributes include alcoholic content, acidity, alkalinity, color intensity, among others. The dataset has 59 instances of the first class, 71 instances of the second class and 48 instances of the third. The classes are labeled as '1', '2' and '3', in order to facilitate a subsequent identification and the validation of the algorithm. The dataset has well defined classes, so that the identification of these classes is performed with relatively little difficulty.
The strategy used in experiments was similar to the previous one, with Iris database. The dataset was divided into two subsets, one with the first six attributes and the other with the seven remaining attributes. The clustering algorithm was applied separately over each one of the subsets and, later, on their representatives.
In the first approach, using traditional SOM, we have used a map size with 11 x 6 neurons in a hexagonal lattice. Training parameters used are $\sigma_{initial} = 2$, $\sigma_{final} = 1$ and 4 epochs in rough phase and $\sigma_{initial} = \sigma_{final} = 1$ and 15 epochs in fine tuning phase.
In the second approach, using proposed strategy, two maps were used, with 9 x 7 and 11 x 6 neurons in local phase, and a map with 11 x 6 neurons in final phase, both plans and hexagonal lattice. Training parameters used in all three maps are $\sigma_{initial} = 2$, $\sigma_{final} = 1$ and 4 epochs in rough phase and $\sigma_{initial} = \sigma_{final} = 1$ and 15 epochs in fine tuning phase. In both

approaches, maps size was automatic defined by SOM Toolbox, considering principal components of data distribution.
The results obtained with the proposed strategy were very near to those of the traditional approach, where all the variables were analyzed simultaneously. The U-matrix obtained with the Wine dataset, using both approaches, is presented in Fig. 3.

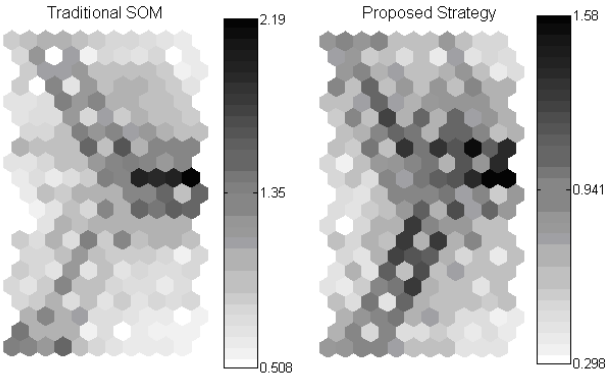


Fig. 3. U-matrix of the SOM algorithm (left) and the proposed strategy (right), using the Wine dataset

Additionally, Fig. 4 shows final self-organizing map with the respective labels that identify each neuron. The numbers represent the class that the neuron belongs to. This is obtained by a voting process on how many elements of the original dataset the neuron is associated to. Fig. 5 presents the same maps labeled to improve the visualization quality. Maps were manually colored based on labels obtained in previous stage.

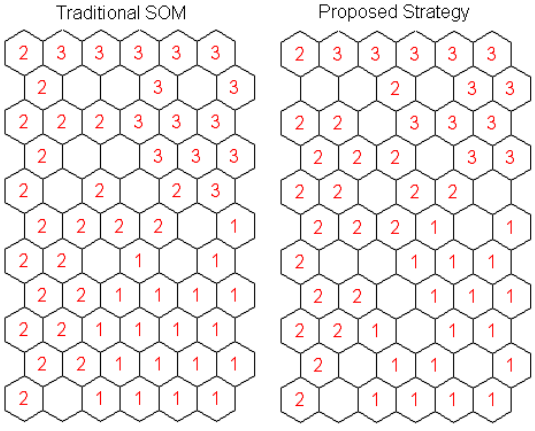


Fig. 4. The self-organizing map, with the respective labels that identify the class of each neuron, for the traditional SOM algorithm (left) and the proposed strategy (right), using the Wine dataset

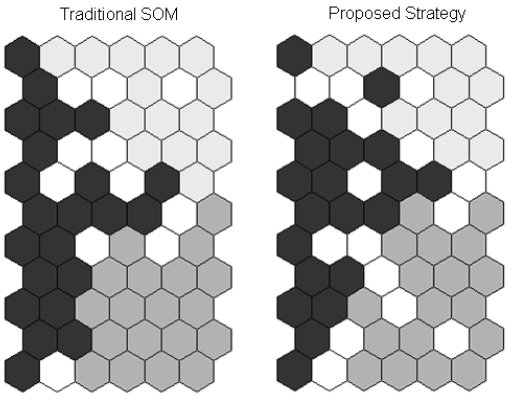


Fig. 5. The manually coloured self-organizing map for the traditional SOM algorithm (left) and the proposed strategy (right), using the Wine dataset.

Table 4 summarizes the average results obtained with traditional SOM and proposed strategy. It presents QE (average quantization error) and TE (topographic error) to Wine dataset. In a similar form of previous section, results obtained with the proposed algorithm were very near to those of traditional methods, using a single dataset. Furthermore, the experiment was replicated using different attribute subsets. The dataset was divided into three, four and five partitions to verify how the partitioning method affects the proposed algorithm performance. In both cases, obtained results were very similar to those presented in Table 4.

Algorithm	QE	TE	Errors	Hits (%)
Traditional SOM	1.8833	0.0169	4	97.8%
Proposed Strategy	2.0967	0.0674	7	96.1%

Table 4. Comparatives missing cluster values betweens traditional SOM and the proposed strategy (Wine Dataset)

In a similar experiment, the Wine dataset was analyzed initially using the traditional K-Means algorithm, and next, using the proposed strategy with the K-Means algorithm. As in previous experiment, the dataset was divided into two subsets, with six and seven attributes, respectively. The K-Means algorithm was applied separately over each one of the subsets and, later, on their representatives.

In traditional approach, the centroids number was defined as $k = 3$, identical to classes number. In distributed approach, the centroids number was defined as $k = 66$ in local stage and as $k = 3$ in central stage. At first stage, the k value was maximized for two reasons: first, a large number of prototypes improve the performance of the algorithm, and second, was used the same number of SOM neurons in previous experiment, with the objective to become more adequate the comparative between both approach. Table 5 summarizes the average results obtained with traditional K-Means and distributed strategy.

Algorithm	Errors	Hits (%)
Traditional K-Means	6	96.6%
Distributed K-Means Strategy	7	96.1%

Table 5. Comparatives missing cluster values betweens traditional K-Means and the proposed strategy (Wine Dataset)

6.3 Wisconsin Breast Cancer Dataset

The Wisconsin Breast Cancer dataset has 699 instances of the cytological analysis of fine needle aspiration of breast tumors. Each instance contains 10 attributes that are computed from a digitized image of a fine needle aspiration of a breast mass. Attributes include radius, texture, perimeter, area, smoothness, compactness, concavity, concave points, symmetry, and fractal dimension. The dataset contain 458 (65.52%) benign instances and 241 (34.48%) malignant instances.

The SOM algorithm was applied to the entire database and to some partitions of the same dataset, according to proposed strategy. The first five attributes were trained in a map and the four remaining attributes were trained separately in another. As before, representatives of both groups were gathered and combined to form a new dataset, which was later clustered using the SOM algorithm again.

In the first experiment, using traditional SOM, was used a 2D hexagonal lattice map with size 22 x 6 neurons. In the second experiment (proposed strategy) two maps with 19 x 7 and 17 x 8 neurons in local phase and a map with 22 x 6 neurons in final phase were used. All SOM were 2D hexagonal lattice maps. As in previous experiments, maps size was automatic defined by SOM Toolbox, based on input data distribution. In both experiments, training parameters used are $\sigma_{initial} = 3$, $\sigma_{final} = 1$ and *trainlen* = 2 epochs in rough phase and $\sigma_{initial} = \sigma_{final} = 1$ and *trainlen* = 15 epochs in fine tuning phase.

Table 6 summarizes the obtained results with traditional SOM and proposed strategy, including QE (average quantization error) and TE (topographic error) to Wisconsin Breast Cancer dataset.

Algorithm	QE	TE	Errors	Hits (%)
Traditional SOM	0.9461	0.0272	17	97.57%
Proposed Strategy	1.0009	0.0615	13	98.14%

Table 6. Comparatives Missing Cluster Values Betweens Traditional SOM and the Proposed Strategy (Wisconsin Breast Cancer Dataset)

Fig. 6 presents the final self-organizing map, which was manually colored to improve the visualization quality. Darker tones represent benign instances and light tones represent malignant instances. As in previous experiment, each neuron is labeled based on amount of instances from original dataset associated with this neuron and privileged information is used only to label the map and verify the algorithm accuracy.

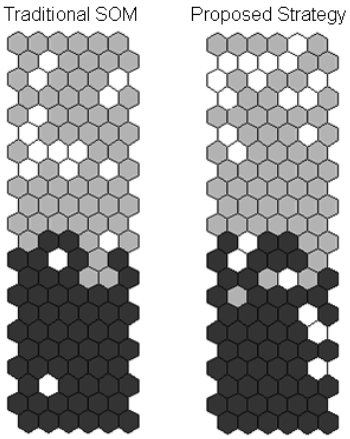


Fig. 6. The manually labelled self-organizing map for the traditional SOM algorithm (left) and the proposed strategy (right), using the Wisconsin Breast Cancer dataset.

6.4 Mushroom Dataset

The Mushroom dataset contain 8124 instances and 22 categorical attributes, with descriptions of hypothetical samples corresponding to several species of gilled mushrooms. Attributes include shape, surface, color, and others. Each species is identified as definitely edible, definitely poisonous, or of unknown edibility and not recommended. This latter class was combined with the poisonous one.

Initially, a pre-processing step was required to convert all categorical data to numeric data, because traditional SOM algorithm handles only numeric data. This processing increases the complexity of the clustering process, since attributes number (dimensionality) is enlarged. After pre-processing, the number of database attributes increased to 117.

The strategy used in experiments was similar to the previous one. In traditional approach, the SOM algorithm was applied to the entire database. However, in the proposed approach, dataset was partitioned into two, three e four subsets. In all experiments, the partitions were carrying out in order that similar attributes were grouped together. For example, 'cap_shape', 'cap_surface' and 'cap_color' attributes were grouped in the same subset.

In the first approach, using traditional SOM, was used a map size with 23 x 19 neurons in a hexagonal lattice. Training parameters used are $\sigma_{initial} = 3$, $\sigma_{final} = 1$ and 1 epoch in rough phase and $\sigma_{initial} = \sigma_{final} = 1$ and 3 epochs in fine tuning phase.

In the second approach, using proposed strategy, the dataset was divided into two subsets, one with the first 49 attributes and the other with the 68 remaining attributes. Two maps were used, with 25 x 18 and 23 x 19 neurons in local phase, and a map with 23 x 19 neurons in final phase, both plans and hexagonal lattice. Training parameters used are $\sigma_{initial} = 4$, $\sigma_{final} = 1$ in first map and $\sigma_{initial} = 3$ and $\sigma_{final} = 1$ in second map, and training are performed using 1 epoch in rough phase and 2 epochs in fine tuning phase. In final map, were used $\sigma_{initial} = \sigma_{final} = 1$ and 1 epoch in rough phase and 2 epochs in fine tuning phase.

Next, the dataset was divided into three subsets, with 49, 33 e 35 attributes respectively. The clustering algorithm was applied separately over each one of the subsets and, later, on their representatives. Three maps were used, with 25 x 18, 25 x 18 and 23 x 19 neurons in local

phase. Training parameters used in local phase were $\sigma_{initial} = 2$, $\sigma_{final} = 1$ and 1 epoch in rough and tuning phase. In final phase, was used a plan and hexagonal lattice map with 25 x 18 neurons. Training parameters used were $\sigma_{initial} = \sigma_{final} = 1$, 1 epoch in rough and 3 epochs in fine tuning phase.
Finally, the dataset was divided into four subsets, with 31, 18, 33 e 35 attributes respectively. The clustering algorithm was applied separately over each one of the subsets and, later, on their representatives. Four maps were used, with 25 x 18, 23 x 19, 25 x 18 and 23 x 19 neurons in local phase, and a map with 23 x 19 neurons in final phase, both plans and hexagonal lattice. Training parameters used in first and third maps are $\sigma_{initial} = 4$, $\sigma_{final} = 1$, 1 epoch in rough and fine tuning phase. In all remain maps, parameters used are $\sigma_{initial} = 4$ e $\sigma_{final} = 1$ and 3 epochs in fine tuning phase.
Table 7 summarizes the average results obtained with Mushroom dataset, using traditional SOM and proposed strategy with two, three and four partitions. The average quantization error (QE) and the topographic error (TE) are presented. Also, are presented the number of errors and hits obtained in the application of the algorithm in a classification task. As in previous section, results obtained with the proposed algorithm were very near to those of traditional methods, using a single dataset.

Algorithm	QE	TE	Errors	Hits (%)
Traditional SOM	5.545	0.042	295	96.37%
Proposed Strategy with 2 partitions	5.859	0.049	292	96.41%
Proposed Strategy with 3 partitions	8.048	0.159	331	95.93%
Proposed Strategy with 4 partitions	5.783	0.062	292	96.41%

Table 7. Comparatives missing cluster values betweens traditional SOM and the proposed strategy (Mushroom Dataset)

One of the main advantages of the proposed strategy in relation to the traditional approach is the data transfer reduction between remotes and central units. Suppose the input data consist of $N \times p$ matrix, where N denotes the number of instances existent in the database and p denotes the data dimensionality. The codebook can be represented as a $k \times p$ matrix, where k denotes the number of prototypes in the codebook and the indexes vector can be represented as a $N \times 1$ matrix.
Therefore, since only the indexes vector and the codebook are transferred from remotes units to central unit, and generally, $k \ll N$, the strategy appears to be sufficiently adequate for application in distributed data clustering, because the traffic among units is reasonably decreased.

Algorithm	Unit1	Unit2	Unit3	Unit4	Total
Traditional SOM with 2 partitions	398,076	552,432	-	-	950,508
Proposed Strategy with 2 partitions	30,174	37,840	-	-	68,014
Traditional SOM with 3 partitions	398,076	268,092	284,340	-	950,508
Proposed Strategy with 3 partitions	30,174	22,974	23,419	-	76,567
Traditional SOM with 4 partitions	251,844	146,232	268,092	284,340	950,508
Proposed Strategy with 4 partitions	22,074	15,990	22,974	23,419	84,457

Table 8. Comparatives data volume transferred betweens traditional SOM and the proposed strategy (Mushroom Dataset)

Table 8 presents a comparative between traditional SOM and the proposed approach, in relation to the data volume transferred, considering two, three and four remote units. Only 1 byte for each attribute was considered, since all the attributes were converted to binary in pre-processing step.

7. Commented Results

The main contribution of this work is to present an alternative strategy for distributed data clustering in vertically partitioned databases using common algorithms. This approach can be performed in a distributed way, executing several instances of the same (or another) algorithm in parallel, each at one site, and combining all results in a central site.
In the first experiment, the Iris dataset was analyzed with proposed strategy using two approaches: first, using SOM in both local and central stages, and second, using SOM in the local stage and K-Means in the central stage. We compared traditional approach with distributed strategy and showed that the very similar results can be obtained in both approaches.
In the second experiment, the Wine dataset was analyzed with traditional SOM and traditional K-Means algorithms and with the proposed strategy using multiple SOM and multiple K-Means instances. The results show that the number of errors in the application of the strategy in a classification task is near to obtained with traditional approaches. Additionally, we carried out other experiments, with 3, 4, 5 and 6 partitions, obtaining similar results.
In the third experiment, the Wisconsin Breast Cancer dataset was analyzed with traditional SOM algorithm and with the proposed strategy using multiple SOM instances. The results show that the number of errors in the application of the strategy in a classification task is near to obtained with traditional approach.
In the fourth experiment, the Mushroom dataset was analyzed with traditional SOM and proposed strategy. We carried out experiments with two, three and four partitions, and results show the efficiency of proposed strategy. Additionally, we analyze the traffic among units and it was showed that using the proposed strategy, the data volume transferred can be efficiently reduced.

8. Conclusion

There is a growing need for effective approaches to analyze data distributed among several sites. This is motivated by the increase in geographically distributed databases and security policies. Merging several parties of a database into a single site is not recommended in some applications, because confidential information can be remounted. Distributed data clustering algorithms are an alternative approach to cluster analysis tasks in distributed databases, because do not require very large databases and reduce inter-sites communication.
Two common algorithms, SOM and K-Means have been widely used in data clustering applications. K-Means is the most used, because is simple and easy of implementation. Self-organizing maps have some advantages in visualization process, including topology preserving mapping, which maps similar data vectors together.

In this work, we propose a novel strategy for data clustering in vertically partitioned databases using a distributed approach that execute typical clustering algorithms (such as self-organizing maps and K-Means) in local units of a distributed database and combine partial results in a central unit. This strategy can be applied to cluster geographically distributed databases, as to avoid excessive data overload from remote units as to maintain the data security and privacy.
We carried out several experimental using UCI databases for demonstration purposes and showed that the proposed strategy, running in a distributed environment, obtained similar results to those with same clustering algorithms, running in traditional approach.
Also, were discussed previously published studies and was presented situations in which for security and privacy reasons or due to high cost of transferring data to a central unit, a central database cannot be recommended. Otherwise, it was demonstrated that privacy-preserving data is guaranteed, since only reference vectors are sent to the central site.
Future research directions will use a mean and residual-vector quantization approach to increase security and data representation.

9. References

Asuncion, A. & Newman, D. J. (2007). *UCI Machine Learning Repository*, available online on: http://www.ics.uci.edu/~mlearn/MLRepository.html, Department of Information and Computer Science, University of California, Irvine, CA.

Berkhin, P. (2006). A Survey of Clustering Data Mining Techniques, In: *Grouping Multidimensional Data: Recent Advances in Clustering*, J. Kogan, C. Nicholas and M. Teboulle (Ed.), pp. 25–71, Springer-Verlag, New York.

Costa, J. A. F. & Andrade Netto, M. L. (2001a). A new tree-structured self-organizing map for data analysis, *Proc. of the Intl. Joint Conf. on Neural Networks*, Washington, DC, Vol. 3, pp. 1931-1936.

Costa, J. A. F. & Andrade Netto, M. L. (2001b). Clustering of complex shaped data sets via Kohonen maps and mathematical morphology, In: *Data Mining and Knowledge Discovery*, B. Dasarathy (Ed.), pp. 16-27, Proceedings of the SPIE, Florida, USA.

Costa, J. A. F. & Andrade Netto, M. L. (2003). Segmentação do SOM Baseada em Particionamento de Grafos. *Proceedings of Brazilian Neural Networks Conference*, pp. 301-308, São Paulo, Brazil, June 2003.

Damian, D.; Orešič, M.; Verheij, E.; Meulman, J.; Friedman, J.; Adourian, A.; Morel, N.; Smilde A. & van der Greef, J. (2007). Applications of a new subspace clustering algorithm (COSA) in medical systems biology, *Metabolomics Journal*, Vol. 3, No. 1, Mar. 2007, pp. 69-77.

Forman, G. & Zhang, B. (2000). Distributed data clustering can be efficient and exact. *SIGKDD Explorations Newsetterl*. Vol. 2, Dec. 2000, pp. 34-38.

Friedman, J. H. & Meulman, J. J. (2004). Clustering objects on subsets of attributes (with discussion), *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, Vol. 66, No. 4, pp. 815-849.

Gonçalves, M.; Andrade Netto, M. L. ; Costa, J. A. F. & Zullo, J. (2008). A new method for unsupervised classification of remotely sensed images using Kohonen self-organizing maps and agglomerative hierarchical clustering methods, *International Journal of Remote Sensing*, Vol. 29, pp. 3171-3207.

Gorgônio, F. L. & Costa, J. A. F. (2008a). Combining Parallel Self-Organizing Maps and K-Means to Cluster Distributed Data, *Proceedings of the 11th IEEE International Conference on Computational Science and Engineering*, CSE'2008, São Paulo, Brazil, 2008, pp. 53-58.

Gorgônio, F. L. & Costa, J. A. F. (2008b). Parallel Self-organizing Maps with Application in Clustering Distributed Data, *Proceedings of the International Joint Conference on Neural Networks*, IEEE World Congress on Computational Intelligence, Hong-Kong, 2008, Vol. 1, pp. 420.

Gorgônio, F. L. (2009). *Uma Arquitetura para Análise de Agrupamentos sobre Bases de Dados Distribuídas*, Tese de Doutorado, Federal University of Rio Grande do Norte, UFRN/PPgEEC, Natal, RN, Brazil.

Haykin, S. (1999). *Neural networks: A comprehensive foundation*, 2nd ed., Macmillan College Publishing Company, New York.

He, Z.; Xu, X. & Deng, S. (2005). Clustering Mixed Numeric and Categorical Data: A Cluster Ensemble Approach, Technical Report, available online on: http://aps.arxiv.org/ftp/cs/papers/0509/0509011.pdf, 2005.

İnan, A.; Kaya, S. V.; Saygın, Y.; Savaş, E.; Hintoğlu; A. A. & Levi, A. (2007). Privacy preserving clustering on horizontally partitioned data, Data & Knowledge Engineering, Vol. 63, No. 3, Dec. 2007, pp. 646-666.

Jagannathan, G.; Pillaipakkamnatt, K. & Wright, R. N. (2006). A New Privacy-Preserving Distributed k-Clustering Algorithm, Proc. of the 2006 SIAM International Conference on Data Mining, pp. 492-496.

Kantardzic, M. (2003). *Data Mining: Concepts, Models, Methods, and Algorithms*, IEEE Press.

Kargupta, H.; Huang, W. ; Sivakumar, K. & Johnson, E. (2001). Distributed Clustering Using Collective Principal Component Analysis, *Knowledge and Information Systems Journal*, Vol. 3, No. 4, May 2001, pp. 422-448.

Kohonen, T. (2001). *Self-Organizing Maps*, 3rd ed., Springer-Verlag, New York.

Lam, C. M.; Zhang, X. F. & Cheung, W. K. (2004). Mining Local Data Sources for Learning Global Cluster Models, *Proceedings of International Conference on Web Intelligence*, Iss. 20-24, Sept. 2004, pp. 748-751.

Murtagh, F. (1995). Interpreting the Kohonen self-organizing feature map using contiguity-constrained clustering, *Pattern Recognition Letters*, Vol. 16 , No. 4, April 1995, pp. 399-408.

Oliveira, S. R. M. & Zaïane, O. R. (2004). Privacy Preservation When Sharing Data For Clustering, Proc. of the International Workshop on Secure Data Management in a Connected World, 2004.

Silva, J. C. & Klusch, M. (2006). Inference in distributed data clustering. *Engineering Applications of Artificial Intelligence*, Vol. 19, pp. 363-369.

Ultsch, A. (1993). Self-Organizing Neural Networks for Visualization and Classification, In: *Information and Classification*, O. Opitz et al. (Ed), pp. 301-306, Springer, Berlin.

Vaidya, J. & Clifton, C. (2003). Privacy-preserving k-means clustering over vertically partitioned data. Proc. of the Ninth ACM SIGKDD Intl. Conf. on Knowledge Discovery and Data Mining, pp. 206-215, ACM, New York, NY.

Vesanto, J. (2000). *Using SOM in Data Mining*, Licentiate's Thesis, Department of Computer Science and Engineering,Helsinki University of Technology, Espoo, Finland.

Vesanto, J. & Alhoniemi, E. (2000). Clustering of the Self-Organizing Map, *IEEE Transactions on Neural Networks*, Vol. 11, No. 3, May 2000, pp. 586–600.

Xu, R. & Wunsch II, D. (2005). Survey of Clustering Algorithms. *IEEE Transaction on Neural Networks*, Vol. 16, No. 3, May 2005, pp. 645-678.

5

Kohonen Maps Combined to K-means in a Two Level Strategy for Time Series ClusteringApplication to Meteorological and Electricity Load data

Khadir M. Tarek, Khdairia Sofiane and Benabbas Farouk
Laboratoire de Gestion Electronique de Documents (LabGED)
University Badji Mokhtar of Annaba, Algeria
khadir@labged.net, khdairia@labged.net, benabbas@labged.net

1. Introduction

Since the start of the computer era, a substantial amount of information and data are stored on numerical form. Automatic classification becomes therefore a very useful tool in order to reduce data dimension and extract maximum knowledge for such configurations (Jajuga *et al.*, 2002).

Data classification is a very important data analysis operation, consisting in regrouping objects of a similar data set into homogenous classes. Two main types of classifications exist: supervised and unsupervised classification. Supervised classification is based on a set of objects L of known classes, called training set, with the main goal being to identify candidate objects into their belonging classes. Where, unsupervised classification consists in partitioning a set of data D into sub-sets of similar attributes called classes or clusters (Halgamuge, 2005). Unsupervised classification is termed *clustering*, and will be so in the remaining of the chapter.

Many linear approaches such as Principal Component Analysis (PCA) (Jolliffe, 2002) and K-means were extensively used for the classification and clustering purposes, with an application to identification of meteorological scenarios in (Reljin *et al.*, 2003). Although PCA proved to be a very useful knowledge extraction technique it suffers from poor visualisation when dealing with complex structure representations of a data sample (Vesanto, 1999).

Nonlinear classification and clustering approaches stand as a strong alternative in order to treat the complexity and visualisation problem inherited from large multidimensional data sets. Self Organising Maps (SOM) or Kohonen maps qualify as a strong, leading, nonlinear approach. In the remaining of this chapter, they will be combined to K-means in order to solve the meteorological and electricity load day type clustering problems.

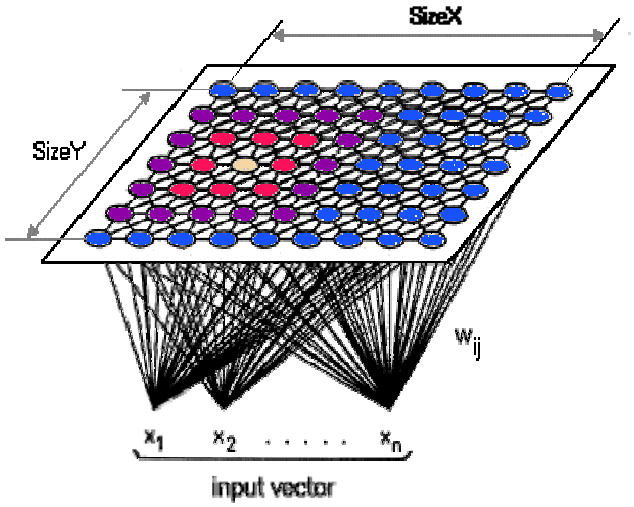


Fig. 1. Two dimensional Kohonen map

2. The Kohonen Self-Organizing Map

The Kohonen self-organizing map (SOM) is an unsupervised classification method, which transforms a set of complex data to one or two dimensional vectors with a simple geometric relationships, and preserving the most important initial data metrics during the display, i.e. the close dataset of the input space will have close representations in the output space and thus will be classified in the same cluster or nearby clusters (Kohonen, 1990, Dreyfus *et al.*, 2004). The self organizing map is suitable for data survey because it has prominent visualization properties; it is also a very effective tool for visualizing and exploring multidimensional data (Himberg, 2000; Vesanto, 1999). SOM has two layers, the input and the Kohonen or output layer, Figure 1.

The network consists in a grid of output nodes connected to the inputs via a set of weights. When presented with the k^{th} input vector P_k ∈R1xn, the network calculates the activation of each node using P_k as:

$$a_{i,j,k} = W_{i,j} P_k \quad (1)$$

where $a_{i,j,k}$ and $W_{i,j}$ are the activation of, and weight (∈ R1×n) connecting P_k to, node i, j respectively. P_k is said to be mapped onto the node with the highest activation. After several inputs have been presented, similar inputs are mapped to the same or adjacent nodes, i.e., within a small neighbourhood. A neighbourhood of size Nc around node i, j is defined as nodes $i \pm Nc$ to $j \pm Nc$. P_k for the current study is formed in two steps.

Each neuron of the topological layer is completely connected to the input layer neurons $Wi = (W_{1i} \ldots W_{ni})$, the weight vectors of these connections form the referent or prototype associated to each neuron, it has the same dimension as the input vectors. In each training step, one sample vector **x** from the input data set is chosen and a similarity measure is

calculated between it and all the weight vectors of the map. The Best-Matching Unit (BMU), is the unit whose weight vector has the greatest similarity with the input sample *P*. The similarity is usually defined by means of a distance measure; typically Euclidian distance. The use of neighbourhood concept introduces the topological constraints in the final SOM geometry.
The weights may or may be not, initialised randomly. In some cases they are initialised around the mean of the inputs as the inputs are all similar and thus restricted to a small portion of the space.
The neurons of competitive networks (Kohonen maps) learn to recognize groups of similar input vectors. Thus, the neuron whose weight vector is closer to the input vector is then updated to be even closer. The result is that the winning neuron is more likely to win the competition next time if similar vector is presented, and less likely to win when a very different input vector is presented. The training stage stops when any of the following conditions are met: the maximum number of epochs is reached, the performance has been minimized to the goal, or maximum amount of time has been exceeded.
During training the inputs are presented one by one and the weights of the triggered node (the node to which the inputs is mapped) and nodes in its neighbourhood are updated as in equation (2).

$$W_{i,j}(m+1) = W_{i,j}(m) + \alpha(m)\left[P_k - W_{i,j}(m)\right] \tag{2}$$

Where a is the adaptation gain, with $0 < a < 1$, and m is the iteration number. This has the effect of increasing the activation of the triggered node and its neighbours. In a single iteration all the inputs are presented and the weights adapted. After several iterations, the neighbourhood size is reduced by one and so on until zero, i.e., the triggered node only is adapted.
The SOM has proved his usefulness for multidimensional dataset clustering treating non-linear problems. The SOM is capable to extracting the statistical properties of time series.

3. The K-Means Clustering Algorithm

The K-means clustering algorithm is a most known vector quantization method; it groups in classes a set of points of the observations space without having any information of particular properties of these groups. Objects are classified as belonging to one of k groups, k chosen a priori. K-means quickly converges to a local minimum of its cost function (Bradley & Fayad, 1998; Kanungo *et al.,* 2002). The aim of K-Means clustering is the optimisation of an objective function that is described by the following equation:

$$l(w,x) = \sum_{x_i} \left\| z_i - w_{x(z_i)} \right\|^2 = \sum_{c} \sum_{x_i \in P_C \cap d} \left\| z_i - w_c \right\|^2 \tag{3}$$

where he expression:

$$I_c = \sum_{z_i \in P_C \bigcap A} \left\| z_i - w_c \right\|^2 \tag{4}$$

represents the local inertia, compared with the referent *Wc* of the training set observations A which are affected to this referent, these observations belong thus to the subset P_C. Inertia I_c is the quantization error obtained when deciding to replace P_C observations by the referent W_c which represents them. The quantity *I (W, X)*, which represents the sum of local inertia I_c is given by:

$$I(W, X) = \sum_c I_c \sum_{\substack{x_i \in A \\ X(x_i = c)}} \left\| z_i - w_c \right\|^2 \tag{5}$$

The K-means algorithm is iterative, where every iteration can be performed in two stages:
Assignment phase: This stage aims to minimize the function *I (W, X)* compared to the assignment function *X* (determining the set of referents).
Minimization phase: The second stage aims optimizing referents in order to representing the best observation points in *p* classes.
The main objective of applying K-means to time series analysis is to identify the different clusters representing the series situation using clustering methods. These methods must provide groups which members are close (have high similarity degree) and well separated. In the clustering process, there are no predefined classes and no examples that would show what kind of desirable relations should be valid among the data, so it is natural to be asked about the validity and quality of the results obtained. Two different sets of validity indices may be used for comparing the results when dealing with K-means: the internal and external criteria. The first indices category quantifies the match between a subjective partition and the idea that there is a good classification (Guérif, 2006) and the most properties commonly searched are compactness and group's separability. Different internal validity indices will be used in the rest of the chapter for meteorological and electricity load clustering, and are summarized in what follows:
***Davies-Bouldin index*:** takes into account both the compactness and the separability between groups, the value of this index is even lower than the clusters are compact and well separated (Davies & Bouldin, 1997). This index favoured hyperspheric groups and it is particularly well adapted for use with the K-means clustering algorithm (Guérif, 2006).
***Silhouette index*:** Kaufman and Rousseeuw (1990) suggest choosing the number of groups k> 2, which gives the greatest value of silhouette.
***Homogeneity and separation*:** homogeneity is calculated as the average distance between each input vector and the centre of the group to which it belongs. The separation is calculated as the average distance between the weighted groups centres (Chen *et al.,* 2002).
***The System Evolution (SE) method*:** Analyzes a dataset as a pseudo thermodynamics system, partition energy $E_p(k)$ denotes the border distance between two closest clusters (called twin-clusters) among the k clusters, while merging energy $E_m(k)$ denotes the average distance between elements in the border region (Wang *et al*., 2007).
***Weighted inter-intra index*:** proceeds with a forward searching and stops at the first mark to the bottom of the index, which indicates the optimal number of groups (Strehl, 2002).
For the external category the validity indices are used:

Rand index and Mirkin metrics. The rand index shows the proportion of pairs object where two partitions are concordant (Guérif, 2006), whereas the Mirkin metrics is defined as the number of edges that exists only in one of two partitions.
Hubert index. Higher values of this index show a large similarity between two groups (Halkidi *et al.,* 2001).

4. A two clustering level approach

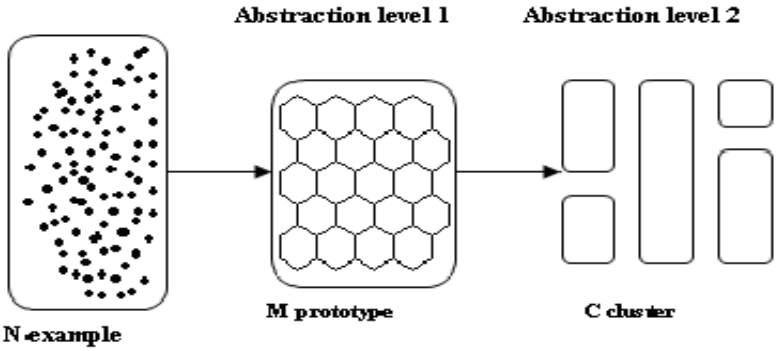


Fig. 2. First abstraction level is obtained by creating a set of prototypes vectors using the SOM. Clustering of the SOM creates the second abstraction level.

The number of prototype vectors resulting from SOM clustering is large especially when dealing with highly multidimensional time series applications. Only one classification level can then be revealing. A high level is interesting because it provides more detailed quality analysis and less compresses the dataset if we summarize all days by representatives of a small class's number (Rousset, 1999). It also can be very difficult to attribute some units of the input vector to a given cluster given by the map. The problem lies in the selection of some clusters border, where a clear distinction between two clusters is impossible. A second clustering stage becomes then useful to remove ambiguity and validate the SOM results.
The approach used in this chapter, is depicted in Fig. 2, the first abstraction level is achieved by creating a set of prototypes using SOM. These prototypes are then clustered in the second abstraction level using K-means clustering algorithm. It was noticed that clustering a large multidimensional time series data using only k-means generates a more computational time than the two-level clustering approach. Another advantage of this approach is the noise reduction (Vesanto & Alhoniemi, 2000), as the prototypes are local averages of the data and therefore less sensitive to random variations than the original data.

5. Application to Meteorological Parameters Clustering

It is extremely important to consider the effect of meteorological conditions on air pollution, because they directly influence the dispersion possibilities of the atmosphere. Severe pollution episodes in the urban environment are not usually attributed to sudden increases in the emission of pollutants, but to certain meteorological conditions which decrease the

capacity of the atmosphere to disperse pollutants (Ziomas et al., 1995). In meteorological studies it is very difficult to represent the data in statistically independent terms because the air pollution depends on all meteorological parameters (Kalkstein, 1991). One of the most interesting applications in the field of meteorological studies is the use of clustering methods in order to extract a representative set of prototypes (clusters) of the meteorological models in an area of interest. This technique has been successfully used in numerous studies such as (Eder et al., 1994). Principal component analysis (PCA) and K-means clustering algorithms have been used in (Reljin et al., 2003) to determine the synoptic weather scenarios. PCA is a powerful linear technique for data reduction (Kwan *et al.,* 2003), but suffers poor visualization as it is not well adapted to represent the datasets complex structure (Laitinen *et al.,* 2002). Recently, and as an alternative tool being used to deal with the complexity of multidimensional data, Kohonen self-organizing maps were used for clustering data in various ecosystems: forest, agriculture, etc. (Recknagel, 2002; Suwardi *et al*., 2007), water quality (Aguilera *et* al., 2001; Tison et al., 2005) and day type identification for electrical load (Khadir et al., 2006). Although the SOM has proved its efficiency in meteorological parameters clustering such as in (Hewitson & Crane, 2002; Cavasos, 2000; Turias et al., 2006), it is difficult to clearly identify the clusters and their borders when the map is very populated.

A two level clustering approach is proposed in this work in order to analyse and identify the meteorological day type for Annaba region in Algeria. In the first stage the SOM was used to reduce the set of prototypes which are then clustered using the K-means clustering algorithm in the second stage. This approach is more powerful than that of a direct clustering in data partitioning and computing time reduction. The correctness of clustering algorithm results is verified using quantitative validation based on two criterions categories (internal and external) and qualitative criteria, these cluster validity indices allowed us to respond to some frequently asked questions such as: "how many clusters are there in the dataset?", "does the resulting clustering scheme fits our data set?", "is there a better partitioning for our dataset?".

5.1. Area of Study and Used Data

Annaba region is located in the Eastern part of Algerian coast (600 km of Algiers), Fig. 3. The town is constituted of a vast plain bordered in the South and West, of a mountainous massive in North, and by the Mediterranean Sea in the East (Mebirouk & Mebirouk –Bendir, 2007). Its basin shaped topography, supports air stagnation and creation of temperature inversions. These situations allow the pollutants accumulation and the rise in concentration rates which results from it. Industry is the main factor causing air quality deterioration; this industrialization has allowed providing the needs of the country and population in iron and steel products, nitrate fertilizers, railway constructions and many other transformation industries. Controversially, it caused a disproportionate urbanization of the town with all its corollaries.

Fig. 3. Location of Annaba region

The dataset used in this study includes 04 meteorological parameters collected for 60 months (1995 to 1999) with a 3 hours expiry, therefore each row of the dataset (unspecified day) is characterized by 32 parameters during the 24 hours. The meteorological parameters which are obtained from the weather station of Annaba are dynamic and thermodynamic air descriptions: the pressure measured in tenth of millibars ; The temperature measured in tenth of °C; The moisture humidity in hundredths and the wind speed measured in nodes. A pre-treatment phase is needed to prepare the data, consisting in noise elimination, error corrections and data standardisation.

5.2. Results of the two Stages Clustering Approach

A. Results of the SOM Map

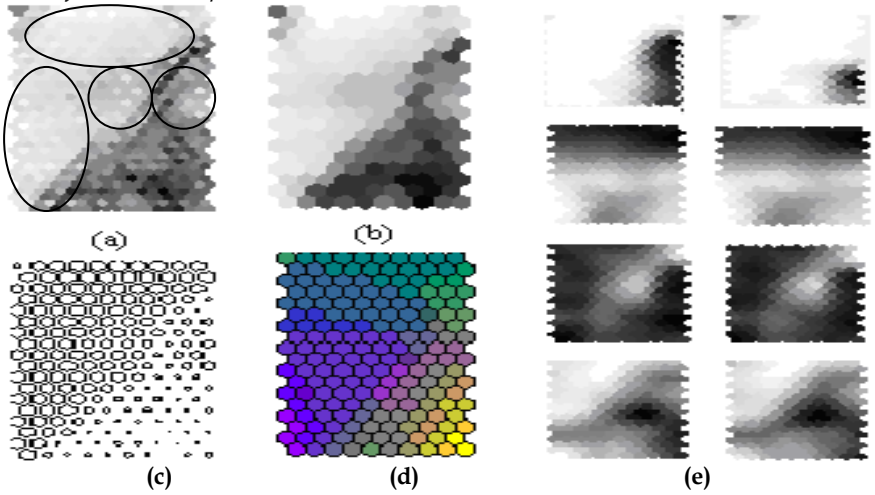


Fig. 4. The U-matrix map, (b and c) are average version of U-matrix, (d) the color coding map, (e) some component plane.

The results obtained from the Kohonen map using the specified dataset are shown in Fig. 4; the maps are connected to adjacent hexagonal nodes with sizes 18×10 by adapting the meteorological situations of the area. There are no explicit rule allowing the choice of Kohonen network node's number, but the principle is that the size should allow easier detection capabilities (Hautaniemi *et al.,* 2003). For this reason different experiments have been done to determine the optimal number of Kohonen units, by changing the number of nodes and checking the performance of each solution. Also different experiments have been done by changing the training parameters in order to determine the appropriate SOM for this dataset. Fig. 4 (a) provides a visualization of the U-matrix which represents a relative measurement of distance between the network coloured units, where the grey colour (shade) of the hexagon indicates the distance measure of the node to its adjacent. More the shade is dark more the distance is large; a cluster which represents similar data vectors can be seen as a clear zone with dark borders.

Fig. 4 (b) and 4 (c) present the average version of the U-matrix, for Fig. 4 (c) the size of each unit of the Kohonen maps is proportional to the average distance to its neighbours. It can be seen, for example, in Fig. 4 (a) that U-matrix provides clustering information of similar units which are presented with some circles on the map. However, the map of U-matrix indicates the situation where the distance measure was not reliable to determine the representative clusters. As reported by (Kiang *et al.,* 2003), it is difficult to visually SOM units when the network is strongly populated. In this case, the decision seems to be difficult and the use only of the Euclidean distance to select the meteorological clusters is not reliable.

To overcome the SOM deficiency in clustering data, a combination of the distance measure and the SOM colour-coding are used. The SOM colour-coding is a method for clustering data, according to their properties (Vesanto, 1999). As shown by Fig. 4 (d), the units which have similar parameters evaluate automatically similar colours of nodes on the grid. Greater distance measures of the network nodes are automatically assigned to different colours and clusters. To select a cluster, we first identify the clusters region based on the discoloration of units. In the situations when colours of nodes are not clear to indicate the differences of the clusters, the distance measures are then used to verify the clusters, Although, it was very difficult to attribute some units to a given group. The problem was the selection of some clusters border, it can be seen that a second clustering stage is useful to remove ambiguity and validate the SOM results.

B. Refining SOM Results by K-means

The K-means clustering algorithm has been applied to group the SOM units with different k-values (the number of clusters in which data are partitioned). Due to the inherent process randomness and because these methods depend on initial centres, the order of the presentation and the geometric properties of the data, a relatively high number of experiments (50 were ran in this study) has to be done and their results checked. The best partitioning for each (k) is selected using the error criteria described by equation (3), also the optimal number of clusters among different values of k is selected according to the validity indices described in section 2. The results of these indices are shown in Fig. 5, .6 and 7.

According to Davies-Bouldin index shown in Fig. 5, a negative peak is noticed at k=6 which indicate the optimal number of clusters proposed by this index. As well as the system evolution method where results values are presented in Fig. 6 indicating that the optimal partition of the dataset is obtained for k=6. The same result is proposed by silhouette and

inter-intra weighted indices shown in Fig. 7. According to the different indices values, the clusters obtained are well separated and homogeneous.

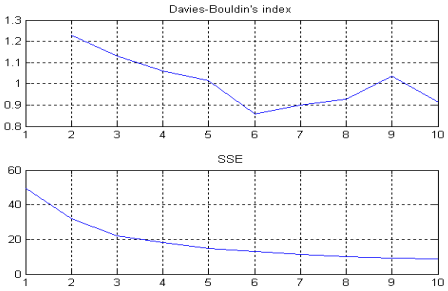


Fig. 5. Davies-Bouldin index and SSE

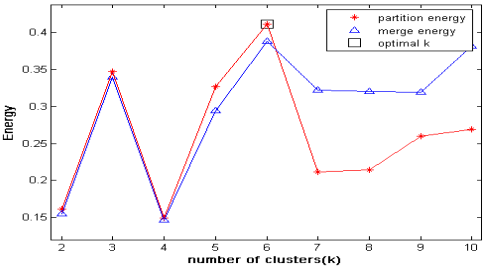


Fig. 6. System Evolution method results

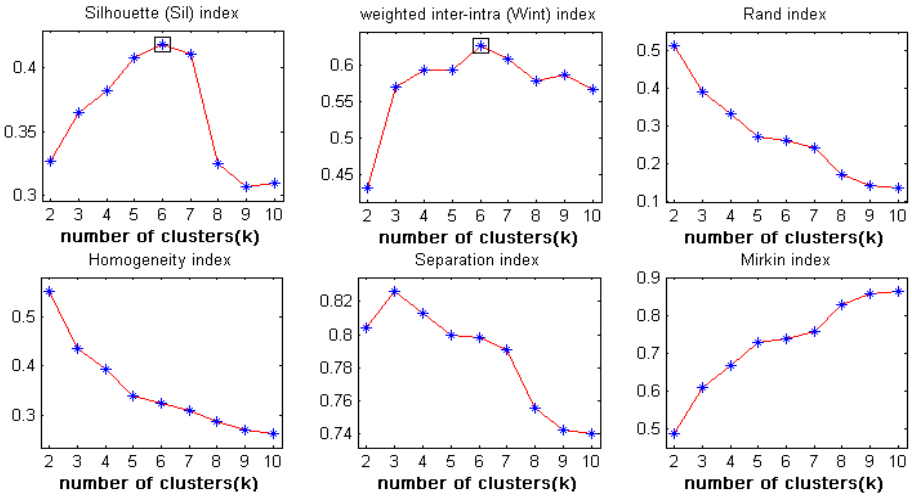


Fig. 7. Internal and external validity indices

The results of the two stage clustering procedure are shown in Fig. 6 and the average meteorological parameters for each cluster are shown in Fig. 9. The cluster C3 is characterized by a steady pressure throughout the 24 day hours, and high temperature which exceeds 25°C during the day and slightly lower in the night, this cluster is also characterized by a high pressure during the night which decrease in the day, the wind speed is very low in the night period and starts increasing during the day, according to the monthly distribution of clusters shown in Fig. 10 this cluster represents the warmer months. The sixth cluster is particularly concentrated in the winter and autumn months and is mainly characterized by a steady pressure and a high wind speed during the day.

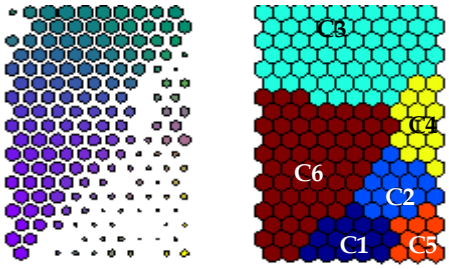


Fig. 8. Second stage clustering results

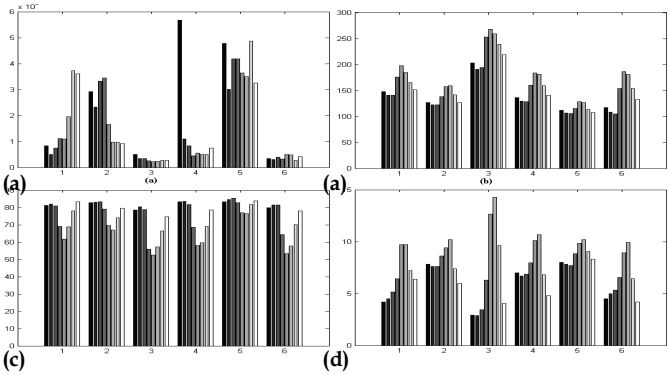


Fig. 9. (a) Pressure mean values, (b) temperature mean values, (c) humidity mean values, (d) wind speed mean values.

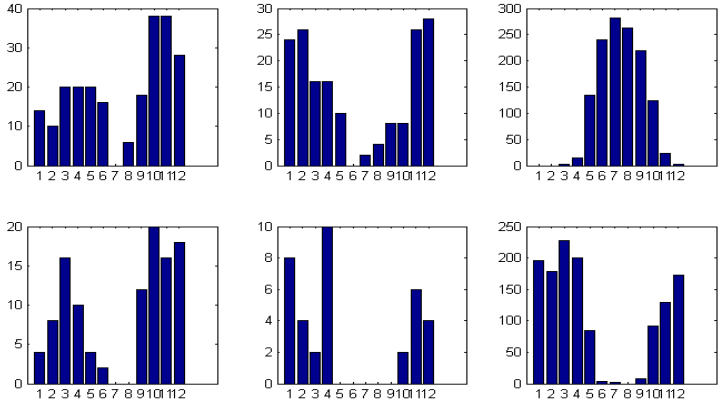


Fig. 10. Monthly distribution of clusters

The fourth cluster parameters are similar to the sixth with a high pressure in the night period and larger wind speed. The fifth cluster is characterized by a high pressure and humidity compared to the other clusters, temperature and wind speed are stable and low all the day hours. The first cluster is almost similar to the fourth with a low pressure at the

beginning of the day, however growing with time, and a lower wind speed values. The second cluster seems to be a sub-cluster of C4 with a small increase in pressure.

6. Day type Identification of Electricity load

Short term electricity load forecasting is nowadays, of paramount importance in order to estimate next day electricity load resulting in energy save and environment protection. Electricity demand is influenced (among other things) by the day of the week, the time of year and special periods and/or days, all of which must be identified prior to modeling. This identification, known as day-type identification, must be included in the modeling stage either by segmenting the data and modeling each day-type separately or by including the day-type as an input. It is proven that the day types or daily consumer's habits for different periods of time, such as working days, weekends, special holidays, etc affect heavily the load shape (Fay, 2004). Different prediction models may then be designed for each day type.

6.1. Overview of Algerian Electricity load

Electrical demand in Algeria from 01/01/2000 to 31/12/2004 is shown in Fig. 11. As can be seen there is an upward trend in the data reflecting increased economic activity over this period.

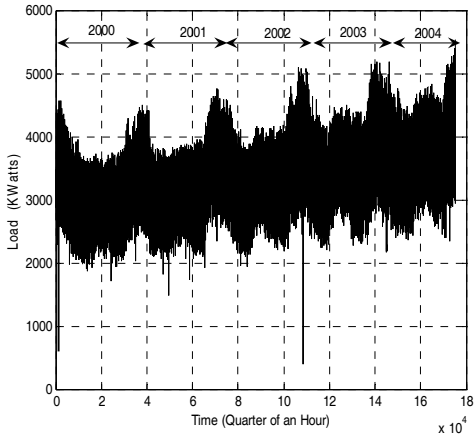


Fig.11. Algerian electricity load 2000-2004.

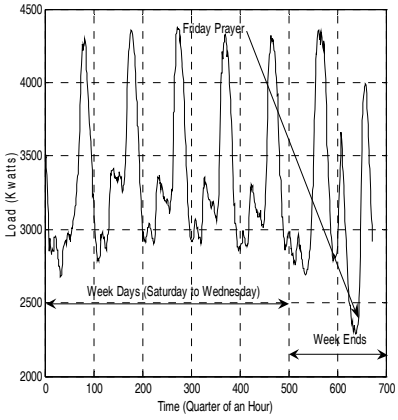


Fig. 12. Weekly load.

Daily load data can be disaggregated into distinct groups (called day-types) each of which has common characteristics. As can be seen in (Fig 12.) there is, for example, an obvious difference between the shapes of the load on a typical weekend day, such as Friday and a working day like Saturday or Sunday due to decreased economic activity and the weekly religious prayer on Friday. Note that in Algeria the weekend is on Thursdays and Fridays. Furthermore, there is a distinct difference between the shape of a typical winter day and summer day.

6.2. Day type identification using Kohonen Map

The existence of several different day-types has been shown by several researchers (Bretschneider *et al.*, 1999; Hsu and yang, 1991; Muller and Petrisch, 1998) However, the level of desegregation in day-type selection is, to a large extent, subjective and dependant on the judgment of the forecaster. As pointed out by (Hubele and Cheng 1990), the application of a separate load forecasting model for different seasons (for example summer, autumn, winter and spring) has the advantage that the models do not need to incorporate seasonal information.

Further desegregation of the load by day of the week (for example Summer Sunday, Winter Sunday, Summer Monday etc.) reduces further the amount of information that the model need incorporate. Such approaches have been implemented successfully by (Srinivasan *et al.*, 1999) and (Mastorocostas *et al.*, 1999), to mention but a few. Where a single model is used for all the data, the day-type information is often incorporated as an additional input (two examples are (Chen *et al.*, 1992) and (Lertpalangsunti and Chan, 1998). In either case the day-types must, however, be identified. The selection of day-types can be guided by analytical techniques. The self-organising feature map or Kohonen map (Kohonen, 1990) would appear ideal for day-type identification as the number and similarity between day-types is not known *a priori*. The Kohonen map can be implemented for day-type identification in several different ways (examples are (Fay and Ringwood, 2003; Hsu and yang, 1991; Muller and Petrisch, 1998) however differences in the results are insignificant in most cases thus the algorithm used by Hsu and Yang (Hsu and yang, 1991) was chosen.

For the present trials, the full years of data from 2003 and 2004 for the region of Algiers (north center and capital of Algeria) were used. The Kohonen map was trained using the following parameters, an initial neighborhood size of $Nc=1$, adaptation gain equal to 0.003, a total number of iteration m=10 and a grid size 18*18 (324) in total.

Initially, the daily load curve is extracted from each day to give a set of load curves that have a minimum value of zero and a maximum value of one (Hsu and yang, 1991).

$$Y'(i)_k = \frac{Y(i)_k - \min Y_k}{\max Y_k - \min Y_k} \quad i = 1,..,24 \tag{6}$$

where $Y'(i)_k$ and $Y(i)_k$ are the i^{th} elements (hour) of the load curve $Y'_k \in R^{1\times 24}$, and actual load $Y_k \in R^{1\times 24}$ of day k respectively. The load curves are then normalised to give them unity length:

$$P(i)_k = \frac{Y^i(i)_k}{\left(\sum_{j=1}^{24} {Y'_k}^2\right)^{1/2}} \quad i = 1,..,24 \tag{7}$$

where $P(i)_k$ is the i^{th} element of P_k. The weights are initialised as:

$$W_{i,j} = \left\| \left[\left(\mu_p(1)\right),, \left(\mu_p(24)\right) \right] + 5\mu \left[\left(\mu_p(1)\right),, \left(\mu_p(24)\right) \right] \right\| \tag{8}$$

where $\mu_p(1)$ and $\rho_p(1)$ are the sample mean and standard deviation of P(i) over all k, u is a uniformly distributed random number in the range -0.5 to 0.5 and Wi,j is normalised to unit length as in (Hsu and yang, 1991). Weight update is then done following Equation (9) repeated below for clarity:

$$W_{i,j}(m+1) = W_{i,j}(m) + \alpha(m)\left[P_k - W_{i,j}(m)\right] \quad (9)$$

Fig 13 shows the triggered nodes identified for the years stating from 2000 until 2004. We notice that the triggered nodes are located in the map (i between 0 and 17) and (j between 10 and 20).

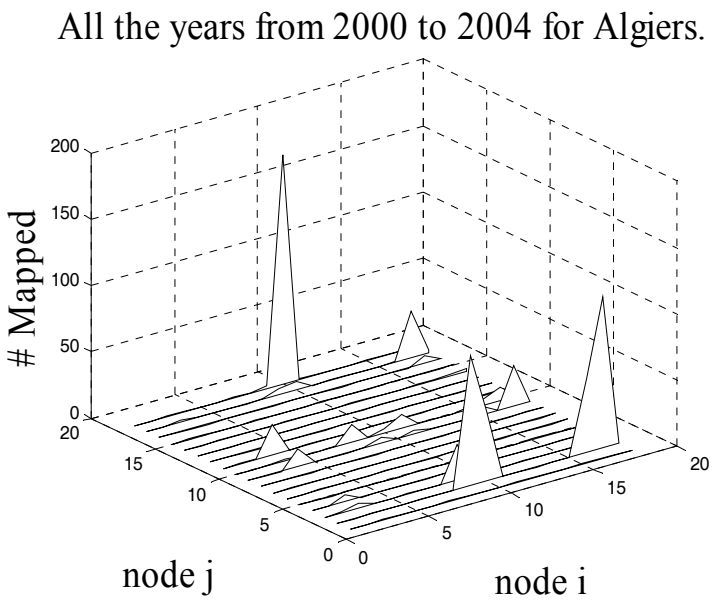


Fig. 13. Kohonen map results for Algiers load

It can be seen, Figure 14, that week days activate roughly the same map nodes where, the weekend activate different nodes for the Algiers load. This is true for Friday which is the weekly prayer occurring from 12 to 2:30. Thursday and Friday are the day of the weekend in Algeria.

Weekdays and weekends however appear differently on the map. The nodes that are triggered from Saturday to Wednesday occupy the same parts of the grid but Thursday and Friday (weekends) loads; trigger different parts of the grid showing the difference between these day types. The figure shows the difference between these days for Algiers load, where the disparate distribution of Fridays appears clearly, and is heavily dependant on seasonal effect.

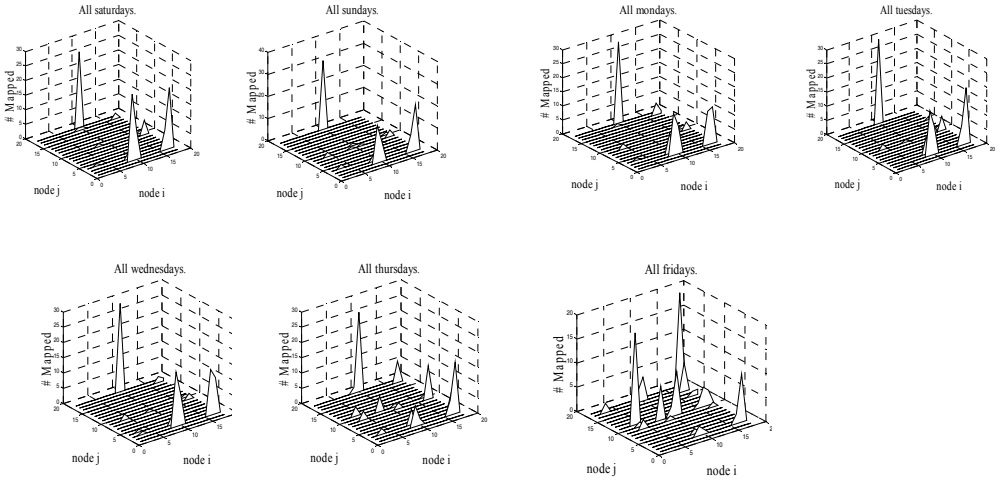


Fig. 14. Nodes triggered for working days(Saturday to Wednesday) and Week days (Thursday and Friday) loads fort he region of Algiers.

The seasonal effect is clearly shown for northern cities, Fig. 15 for Algiers where peaks appear along the longitudinal axe of the SOM with respect to monthly (seasonal) load. As for southern cities, minor seasonal effect is noticed. As can be seen in Fig. 9, the monthly SOM representation shows common peak for all months with a second peak appearing from May to august. The number of visually identified clusters may be numbered as 8 or 9 clusters.

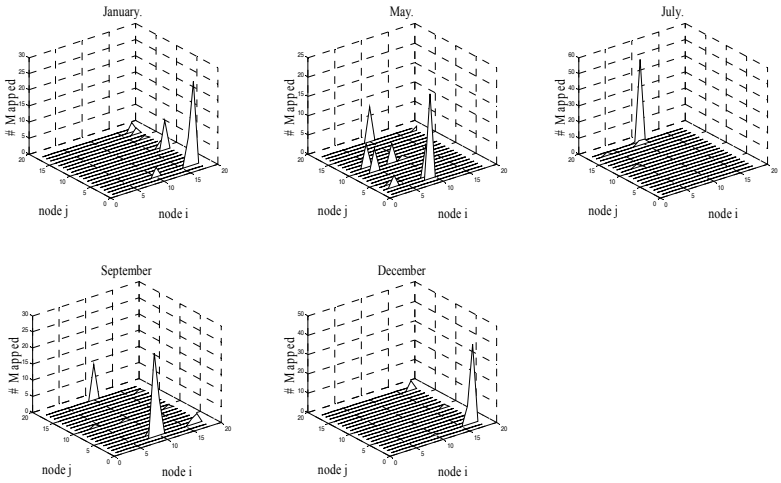


Fig. 15. Seasonal day-type identification for Algiers.

The K-mean algorithm is performed on the output obtained by Kohonen map in order to better define boundaries between clusters, and thus defining clearly the cluster number. The selection of the adequate cluster's number is accomplished using the Davies- Bouldin index defined earlier, and given in the following equations:

$$DB = \frac{1}{k}\sum_{i=1}^{k} R_i \; with \; R_i = max R_{ij} \;\; j = 1, \dots n \; and \; i \neq j \tag{10}$$

k is the number of clusters and

$$R_{ij} = \frac{\left(s(C_i) + \; s(C_j)\right)}{\sigma(C_i \, , C_j)} \tag{11}$$

where $s : C \rightarrow R$ measures the scatter within a cluster and $\sigma : CxC \rightarrow R$ is a cluster to cluster distance measure.

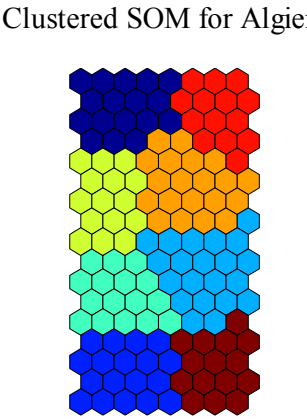


Fig. 16. Definite clusters identified for Algiers load

This clustering procedure aims to find internally compact spherical clusters which are widely separated.
As shown in Fig. 16, the number of clusters is found to be 8 with a value of *DB=0.8788*.

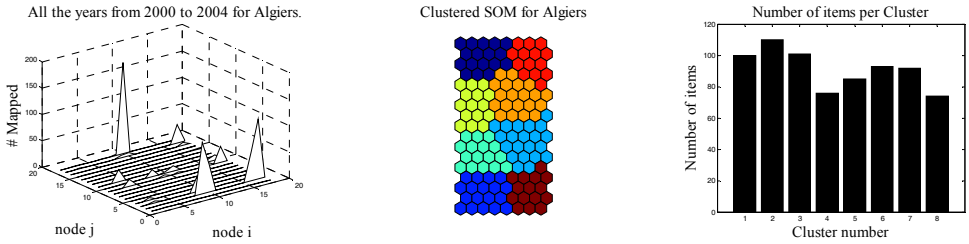


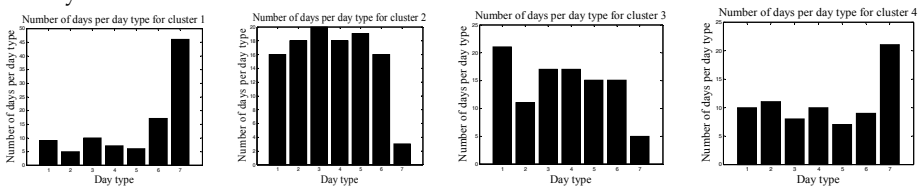
Fig 17. Kohonen map, clustered SOM with k-means and Number of items per cluster

Fig. 17 summarizes the Kohonen map, the clustered SOM and the number of items per cluster for Algiers's load. It can be also deducted that some clusters are dominant in terms of number of days, e.g., C3 .
Table 1 shows the weekly clusters distribution. Where, for example C1 contains a majority of Fridays. This includes that the cluster represent weekends at certain season.

Region	Algiers						
Day type	Sat	Sun	Mon	Tue	Wed	Thu	Frid
C1	9	5	10	7	6	17	46
C2	16	18	20	18	19	16	3
C3	21	11	17	17	15	15	5
C4	10	11	8	10	7	9	21
C5	13	15	12	15	15	12	3
C6	8	12	13	9	11	18	22
C7	18	19	13	14	17	9	2
C8	9	13	11	14	15	9	3

Table 1 Weekly distribution of cluster for Algiers load

Fig. 18 shows the weekly and monthly distribution of clusters. Detailed content of each cluster in terms of day types and number of days is also shown. For example Table 1, shows that C1 contains a majority of Fridays, which makes it a class containing weekends and bank holidays.



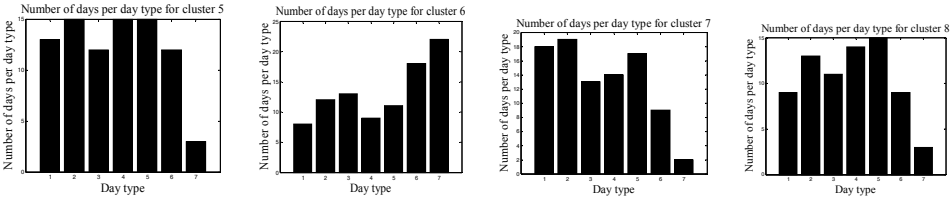


Fig 18. Weekly distribution of cluster

7. Conclusion

Time series analysis using Kohonen maps, allows a rough visual identification of the different existing classes. The K-Means algorithm comes as a complement for better class clustering and a clear frontiers definition when validated using different types of indices. Using a two stage clustering procedure seems to be more efficient than a direct clustering approach involving only SOM or K-means algorithms from an applicative and results view points. The obtained classification is also more compact as it merges neighbouring clusters into one.

Different clusters have been identified with clear borders definition for both day type identification along with a comprehensive analysis for constituents of each cluster in terms of size, day types and seasonal effects for meteorological and electricity load with, respectively, six and eight identified clusters. The results obtained may then be used to design prediction multi-model systems according to the number and the nature of each cluster (data type). Such approach may be more advantageous and can improve the results of a unique global predictor or classifier.

Acknowledgments

This work was supported by the TASSILI program n° 07 MDU 714.

8. References

Aguilera, P.A., Frenich, A.G., Torres, J.A., Castro, H., Vidal, J.L.M. and Canton, M. (2001). Application of the Kohonen neural network in coastal water management: methodological development for the assessment and prediction of water quality. *Water Res. 35, 4053–4062.*

Annas, S., Kanai T. and Koyama, S. (2007). Principal Component Analysis And Self Organizing Map For Visualizing And Classifying Fire Risks In Forest regions, *Agricultural information research, 16(2), 44-51.*

Bradley, P. and Fayyad, U. (1998). Refining initial points for K-means clustering. *International Conference on Machine Learning (ICML-98*), 91-99.

Bretschneider, P., Rauschenbach, T., and Wernstedt, J., (1999). Forecast using an adaptive fuzzy classification algorithm for load," *6th European Congress on Intelligent Techniques and Soft Computing, 3, 1916-1919.*

Cavazos, T. (2000). Using self-organizing maps to investigate extreme climate events: an application to wintertime precipitation in the balkans, *Journal of climate, 13, 1718–1732.*

Chen, G., Jaradat, S.A. and Banerjee, N. (2002). Evaluation and comparison of clustering algorithms in anglyzing es cell gene expression data, *Statistica Sinica* 12, 241-262.

Chen, S.T. Yu, D.C. Moghaddamjo, A.R. (1992). Weather sensitive short-term load forecasting using non-fully connected artificial neural network, *IEEE Transactions on Power Systems, 7 (3), 1098-1104.*

Davies, D.L. and Bouldin, D.W. (1997). A Cluster Separation Measure. *IEEE Transactions On Pattern Analysis And Machine Intelegence, PAMI 1(2):224-227.*

Dreyfus G. , Martinez, J.M., Samuelides, M., Gordon, M.B., Badran, F., Thiria, S. and Hérault, L. (2004). *Reseaux De Neurones : Méthodologie Et Application*, Eyrolles ISBN : 2-212-11464-8, France.

Eder, B.K., Davis, J.M. and Bloomfield, P. (1994). An automated classification scheme designed to better elucidate the dependence of on meteorology. *Journal of Applied Meteorology, 33, 1182–1199.*

Fay, D., Ringwood, J.V., Condon, M., and Kelly, M. (2003). 24-hour electrical load data -a sequential or partitioned time" series? Journal of Neurocomputing, 55(3-4), 469-498.

Fay, D. (2004). *A strategy for short-term load forecasting in Ireland*, Ph.D Thesis, Dept. of Electronic Engineering, Dublin City University, Ireland.

Guérif, S. (2006). *Réduction de dimension en Apprentissage Numérique non supervisé,* Ph.D thesis, Université Paris 13, France.

HALGAMUGE, S.K. (2005*). CLASSIFICATION AND CLUSTERING FOR KNOWLEDGE DISCOVERY (STUDIES IN COMPUTATIONAL INTELLIGENCE),* WANG, L. (EDS), SPRINGER, 3540260730, THE NEDERLANDS.

Halkidi, M., Batistakis, Y. and Vazirgiannis. M. (2001). On Clustering Validation Techniques. *Intelligent Information Systems Journal, 17(2-3): 107-145.*

Hautaniemi, S. Yli-Harja, O., Astola, J., Kauraniemi, P., Kallioniemi, A., Wolf, M., Ruiz, J,. Mousses S. and kallioniemi, O. (2003). Analysis and visualisation of gene expression microarray data in human cancer using self-organizing maps. *Machine Learning, 52, 45-66.*

Hewitson, B. C. and Crane, R. G. (2002). Self-organizing maps: applications to synoptic climatology, *Climate Research, 22, 13–26.*

Himberg, A. (2000). SOM Based Cluster Visualization and Its Application for False Coloring, *Proceedings of International Joint Conference on Neural Networks (IJCNN), 3, 587-592.*

Hsu, Y.Y. and Yang, C.C. (1991). Design of artificial neural networks for short-term load forecasting Part I: Self-organising feature maps for day type identification, *IEEE Proceedings-C, 138(5), pp 407-413.*

Hubele, N.F. and Cheng C.S., (1990). Identification of seasonal short-term forecasting models using statistical decision functions, *IEEE Transactions on Power Systems, 5 (1), 40-45.*

Jajuga, K., Sokolowski, A and Bock, H.H, (2002). *Classification, Clustering, and Data Analysis: Recent Advances and Applications (Studies in Classification, Data Analysis, and Knowledge Organization)*, Springer, ISBN : 354043691X, Berlin.

JOLLIFFE, I.T., (2002). *PRINCIPAL COMPONENT ANALYSIS, SPRINGER*, 0387954422, NEW YORK.

Kalkstein, L.S. (1991). A new approach to evaluate the impact of climate on human mortality. *Environmental Health Perspectives 96, 145–150.*

Kanungo, T., Mount, D. M., Netanyahu, N., Piatko, C., Silverman, R. and Wu, A. (2002). An efficient k-means clustering algorithm: analysis and implementation. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 24(7), 881-892*.

Kaufman L. and Rousseeuw P.J. (1990). *Finding Groups in Data: An Introduction to Cluster Analysis*, Wiley.

Khadir, M.T., Fay, D. and Boughrira, A. (2006). Day Type Identification for Algerian Electricity Load using Kohonen Maps, *Transaction on engineering, computing and technology 15, 296-300.*

Kiang, M.Y., Hu, M.Y. and Fisher, D.M. (2004). An extended self organizing map network for market segmentation a telecommunication example. *Decision Support Systems, DECSUP-11061-12.*

Kohonen , T. (1990). The self-organising map, *Proceedings IEEE, 78 (9). 1464-1480.*

Laitinen, N., Ranatanen, J., Laine, S., Antikainen, O., Rasanen, E., Airaksinen, S. and Yliruusi, J. (2002). Visualization of particle size and shape distributions using self-organizing maps. *Chemometrics and intelligent laboratory systems, 62, 47-60.*

Lertpalangsunti N., and Chan C.W., (1998). An architectural framework for the construction of hybrid intelligent forecasting systems: application for electricity demand prediction, *Engineering Applications of Artificial Intelligence, 11, pp 549-565.*

Mastorocotas P.A., Theocharis, J.B. and Bakirtzis, A.G. (1999). Fuzzy modelling for short term load forecasting using the orthogonal least squares method, *IEEE Transactions on Power Systems, 14 (1), 29-35.*

Mebirouk H., and Mebirouk-Bendir F. (2007). Principaux acteurs de la pollution dans l'agglomération de annaba. Effets et développements, *Colloque International sur l'Eau et l'Environnemen*t, Alger.

Muller, H. Petrisch, G. (1998*). Energy and load forecasting by fuzzy neural networks.* In: H. Jurgen, H.J. Zimmermann eds., Proceedings, European Congress on Intelligent Techniques and Soft Computing, Aachen, Germany, September. Aachen: Elite foundation, 1925-1929.

Recknagel, F. (Ed.), (2002). *Ecological informatics: understanding ecology by biologically-inspired computation*. Springer, Berlin.

Reljin, I., Reljin, B. and Jovanovi, G. (2003). Clustering and mapping spatial-temporal datasets using som neural networks , *Journal Of Automatic Control, 13(1), 55-60.*

Rousset, P. (1999). *Applications des algorithmes d'auto-organisation à la classification et à la prévision.* Ph.D thesis, University of Paris I, France.

Srinivasan D., Tan S.S., and Chang, E.K. (1999). Parallel neural network-fuzzy expert system for short-term load forecasting: system implementation and performance evaluation, *IEEE Transactions on Power Systems, 14(3), 1100-1106.*

Strehl. A. (2002). *Relationship-based Clustering and Cluster Ensembles for High-dimensional Data Mining*. Ph.D thesis, The University of Texas at Austin.

Tison, J., Park, Y.-S., Coste, J., Wasson, M..G., Ector , L., Rimet, F. and Delmas, F. (2005). Typology of diatom communities and the influence of hydro-ecoregions: A study on the French hydro system scale, *Water Research 39- 3177–3188.*

Turias, I.J., Gonzalez, F.J., Martin, M.L. and Galindo, P.L. (2006). A competitive neural network approach for meteorological situation clustering, *Atmospheric Environment, 40, 532–541.*

Vesanto, J. (1999). SOM-Based Data Visualization Methods, *Intelligent Data Analysis, 3(2), 111-126.*

Vesanto J. and Alhoniemi , E. (2000). Clustering of the Self-Organizing Map, *IEEE Transactions on Neural Networks, special issue on data mining, 11(3), 586−600.*

Wang, K., Zhang, J., Zhang H. and Guo, T. (2007). *Estimating the Number of Clusters via System Evolution for Cluster Analysis of Gene Expression Data*, Technical report. Xidian University, P. R. China.

Xu, k.C.R. andHaynes, L. (2001). A new data clustering and its applications .*Proceeding of SPIE-the international society for optical engineering, 4384, 1-5.*

Ziomas, I.C., Melas, D., Zerefos, C.S. and Bais, A.F. (1995). Forecasting peak pollutant levels from meteorological variables. *Atmospheric Environment 29, 3703–3711.*

6

Visual-Interactive Analysis With Self-Organizing Maps - Advances and Research Challenges

Tobias Schreck
Technische Universitaet Darmstadt, Computer Science Department, Interactive Graphics Systems Group
Germany

Abstract
Based on the Self-Organizing Map (SOM) algorithm, development of effective solutions for visual analysis and retrieval in complex data is possible. Example application domains include retrieval in multimedia data bases, and analysis in financial, text, and general high-dimensional data sets. While early work defined basic concepts for data representation and visual mappings for SOM-based analysis, recent work contributed advanced visual representations of the output of the SOM algorithm, and explored innovative application concepts.
In this article, we review a selection of classic and more recent approaches to SOM-based visual analysis. We argue that important improvements have been achieved which allow effective visual representation and interaction with the output of the SOM algorithm. We identify promising directions for future research, which will support new application areas and provide additional advanced visualization approaches.

1. Introduction

Driven by technological advances in data acquisition, processing, storage and dissemination, increasingly large and complex data sets become available. Important fields in which huge amounts of data arise include the Multimedia, Science and Engineering, and Business domains. In Multimedia, digitization of existing and authoring of new content contribute toward formation of large repositories of digital audio-visual material. In Science and Engineering, vast amounts of primary research data arise in experiments, simulation and phenomena observation. And in the Business domain, a wealth of data arising from customer transactions, monitoring of productive processes etc. is captured by many companies. While *storage technology* allows to persistently save much of these data volumes, it is less obvious how *automatic analysis* and *interactive access* methods can support retrieval and discovery of interesting and useful information therein.
To effectively deal with growing amounts of complex data, besides application of purely automatic solutions, a promising approach is to apply visual-interactive data reduction and exploration techniques (Hinneburg et al. (1999)). The general idea here is to abstract large data sets to a condensed representation which captures important aspects in the data, abstracting from details and allowing better navigation and interpretation by the user. The *Self-Organizing Map* (SOM) (Kohonen (2001)) algorithm, to date applied on many different data types, has proven to be a very suited algorithm to this end. SOM has a strong disposition for visual cluster anal-

ysis, as it not only provides the data reduction, but also a spatialization of cluster prototypes forming a baseline for visualization and interaction with the data.
In this article, we survey applications of the SOM algorithm for visually supported retrieval and analysis tasks in a variety of data domains. Our survey is focused on two aspects in SOM-based data analysis. First, we aim to cover a range of different approaches to *visualization* of the output of the SOM algorithm, as visualization often forms the key interface between the algorithm output and the user. Second, we aim to cover a range of different *application* types, illustrating the principal applicability of the algorithm on any data for which a meaningful vector representation can be defined. The choice of examples presented is done to survey both classic, well-established visualization approaches and application types, as well as more experimental and innovative ones. The goal is to motivate the wide applicability of current SOM methodology, and identification of promising future research options leading to further advanced SOM visualizations and application concepts.
The remainder of this article is structured as follows. Section 2 discusses fundamental options for training of Self-Organizing Maps and for their visualization. In Sections 3 and 4, we survey a selection of SOM-based visualization approaches and application concepts, illustrating progress made over basic SOM visualization and application concepts. Section 5 gives an assessment of the surveyed techniques, identifying promising directions for future development of SOM visualization and application fields. Finally, Section 6 concludes.

2. Background

The Self-Organizing Map algorithm is one of the most popular visual cluster analysis algorithms. Its applicability on a broad range of data types has been empirically demonstrated by an abundance of research papers to date. Visualization of SOM output plays a key role in many successful applications of SOM. In Section 2.1, we recall the main issues in training SOMs from complex data, and in Section 2.2, basic options for visual analysis of SOM output are discussed. We also recall results from Information Visualization relevant to SOM visualization in Section 2.3.

2.1 Training of Self-Organizing Maps

The Self-Organizing Map algorithm is a scheme for training a neural network to represent a distribution of high-dimensional input data. A low-dimensional (usually, 2-dimensional) grid of reference vectors is learned from the input data by means of competitive, iterative adjustment of reference vectors to the input data space. Effectively, the SOM is a combined vector quantization and projection algorithm, as (a) many input records are represented by a fixed number of reference vectors, and (b) the reference vectors representing the input data are given a topological ordering by the SOM grid. The SOM yields (a) a clustering of the data and (b) approximately preserves the topology of the data points from the input space, and is therefore especially useful for data visualization and exploration purposes. We here do not recall the SOM algorithm details but instead refer to Kohonen's monograph (Kohonen (2001)).
Application of SOM on a data set requires two main prerequisites: A *vector representation* of the data to be analyzed (usually called Feature Vector, or descriptor), and the *definition of a set of training parameters*. If the data elements are already characterized by application-specific numeric attributes, then these can be used as a descriptor. In case of non-standard data, e.g., multimedia objects, a feature extraction step has to be performed. The latter usually involves heuristics in the definition of the features (Duda et al. (2001)), and can be optionally supported by methods of feature selection (Liu & Motoda (2007)). After the vector description has been

obtained, a number of SOM parameters needs to be set by the user. Main parameters include the resolution and topology of the prototype network, the learning rate and training kernel, and length of the training run. Note that both descriptor extraction and SOM parameterization are non-trivial tasks usually requiring experience of the user both in general data analysis as well as in the specific data domain to be addressed.

2.2 Basic Visual Analysis of SOM Output

The output of the SOM is a network of prototype vectors representing the input data set. Contrary to the output of other cluster algorithms such as e.g., k-means, the SOM output allows straightforward visualization. Usually, the SOM network structure is exploited to map the prototype vectors to the visual variable *position*, while other visual variables such as *shape* and *color* are used to encode properties of the prototype vectors and their relationships regarding the data vectors. Vesanto (1999) distinguishes three SOM-analytic use cases and supporting visualization approaches:

2.2.1 SOM Cluster Structure

Here the task is to assess the structure of the SOM output in terms of the relation between the prototype vectors, usually quantified by their relative vector distances. Distances between reference vectors can be visualized directly on the SOM network by gray- or color-coding (U-Matrix visualization) or by glyph-based approaches scaling e.g., the size of SOM prototype glyphs. Furthermore, the SOM prototype vectors can be compared for similarity by means of subsequent projection methods such as Multidimensional Scaling. Both techniques may support the user in discriminating between regions of the SOM.

2.2.2 Prototype Vector Analysis

Here, a detailed analysis of the properties of prototype vectors in terms of individual vector dimensions is aimed at. The spread of dimensions over the SOM network can be visualized by color-coding of single selected dimensions in form of so-called component planes. Multiple component planes can be visualized simultaneously by arranging them in a matrix layout. Besides the spread of component values, the contribution of selected components toward distances between cluster prototypes can be visualized by color-coding. An important analysis goal here is to understand the correlation between the cluster structure and individual components.

2.2.3 Cluster Structure and Data Distribution

Here, the distribution of data elements underlying the SOM analysis, or new data elements to be discussed in terms of an existing SOM structure, is considered. Single data elements can be positioned on the SOM by simple marks indicating the best matching unit, or response surfaces indicating the distance between the sample to all SOM prototype vectors by means of color-coding. Density histograms visualize the aggregated distribution of many data elements over the SOM by color-coding or glyph size as basic visualization options. Density itself can be estimated in the nearest neighbor sense, but also voting-based approaches forming smoothed density histograms are possible (Pampalk et al. (2002)).

2.2.4 Background Summary

These approaches comprise some of the most fundamental SOM visualization options, and are specifically useful if the vector components can be meaningfully interpreted by the user. In

case of more abstract vector components, which often is the case with derived Feature Vectors, other forms of visualization of represented data can be useful. *Thumbnail Maps* use a visual representation of selected data elements (if available) to visualize the spread to data samples over the SOM.

2.3 SOM and Information Visualization

Information Visualization is concerned with the visual representation and interactive exploration of information spaces that are not inherently spatial in nature. The goal is to foster insight into the data, and stimulate interactive visual data analysis (Ware (2004); Card et al. (1999); Spence (2006)). To date, SOM has been applied for data preprocessing in many Information Visualization systems. Important interaction and visualization concepts developed in Information Visualization are useful for application in SOM-based data analysis.
The ordered layout of data items is an important principle applied in many Information Visualization solutions, and often, Self-Organizing Maps are employed to provide ordering of elements of a data set. Besides direct visualization of the SOM grid as discussed above, coordinated multiple views have been developed which allow deeper exploration of data characteristics of the SOM output and represented data (Guo et al. (2006)). Specific interaction concepts have been developed in Information Visualization for user navigation in dense data spaces (Card et al. (1999)). Distortion techniques adapt the display such to visualize detail information at a focus point set by the user, and compressing the surrounding context information by data reduction or visual aggregation. Dynamic queries allow to interactively filter the data to instantly show data selections matching a user query. Visualization techniques developed for high-dimensional data such as Parallel Coordinates (Inselberg & Dimsdale (1990)), Glyph-based approaches (Ward (2002)) or Pixel-oriented techniques (Keim et al. (1995)) can be used for visualization of high-dimensional SOM prototype vectors.

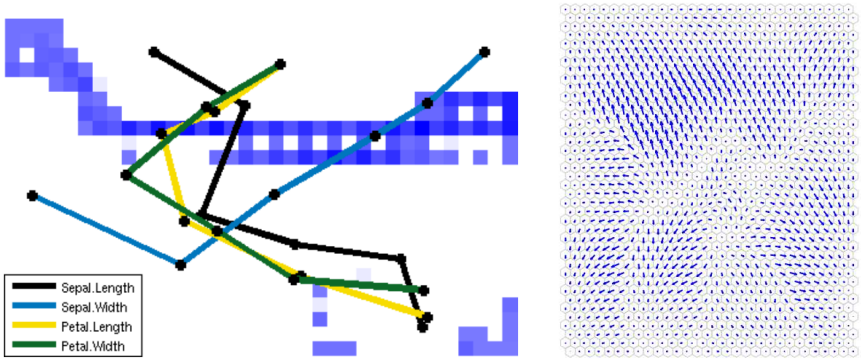
3. Approaches to Enhanced SOM Visualization

In this section, we recall several works we consider innovative on the side of visual SOM support, before we address innovative application domains in the next section. We note that while we made a distinction between visualization and application concept contribution, that distinction is not always clear cut, as often, novel visualization approaches foster new application concepts and vice versa.

3.1 Structural Enhancement for Component Planes

The consideration of characteristics of SOM reference vector components is among the most basic and important SOM analysis tasks (cf. also Section 2.2). Component planes and arrays thereof are a straightforward visualization for the spread of component values over the SOM lattice. However, the user needs to switch back and forth between multiple components to arrive at an understanding of multidimensional component spread, a task which incurs considerable cognitive load. Rauber in Neumayer et al. (2007) proposed to overcome this problem by jointly plotting characteristics of the spread of individual components. Specifically, by means of quantized component planes, path lines indicating the direction of increase or decrease of component values are calculated. Together with appropriate layout adjustments, multiple such path lines are overlaid over a baseline Umatrix visualization. The resulting visualization, called metro visualization, gives an aggregate view over several structural component spreads over the baseline SOM lattice. As it captures multiple component spread patterns, it reduces cognitive user load as no switching between component plane plots is required. Also, the

extracted path lines are an abstraction of the overall information contained in the component plane plots. Detail information is abstracted in favor of overall fundamental information. This in turn may help the user in correlating the individual components. Figure 1 (a) illustrates the concept.



(a) Structural component plane analysis (b) Vector field display

Fig. 1. (a) The so-called metro visualization overlays structural component spread paths over a baseline Umatrix visualization (Figure © 2007 IEEE, taken with permission from Neumayer et al. (2007)). (b) The vector field enhancement indicates the direction of attraction for each SOM cell on the lattice. The resulting visualization is useful for visual evaluation of overall cluster structure given by the SOM (Figure © 2005 IEEE, taken with permission from Polzlbauer et al. (2005)).

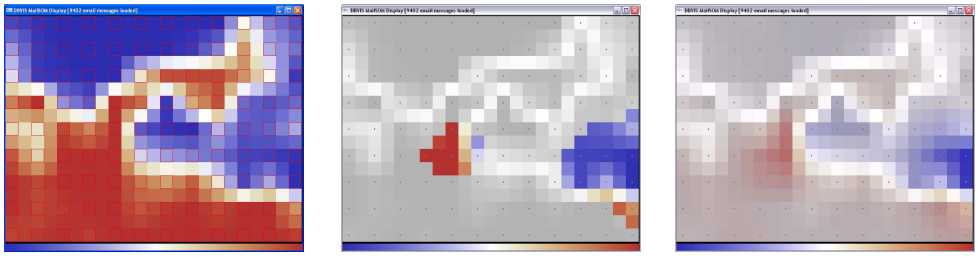
3.2 Vector Fields for Enhanced Umatrix Visualization

Another key analysis task in visual SOM analysis is determination of cluster structure by the user. As the SOM algorithm not directly returns a measure for the number or boundaries of data clusters, this is a key interpretation which the user has to perform. While automatic analysis may help during SOM post-processing (e.g, by applying a post processing cluster algorithm on the SOM prototypes), often the Umatrix visualization is interactively employed to this end. However, the Umatrix in its standard form considers only distances between adjacent SOM reference vectors. A more elaborate visualization to support interactive cluster analysis was proposed in Polzlbauer et al. (2005). For each prototype, the direction toward the most similar map area is evaluated by integration over a set of neighboring prototypes; that direction is then represented in form of a vector. If the direction vectors are plotted on the SOM grid, the resulting gradient vector field visualization gives visual hints to the user which may help in identification of the SOM cluster structure. The method comprises a user-settable parameter for the definition of the neighborhood size over which the similarity is evaluated. Figure 1 (b) illustrates a gradient vector field for a SOM lattice.

3.3 Bivariate Color Maps

Email nowadays is one of the most important means of communication, and many people maintain growing personal email archives constituting large information collections. Besides

the possibility to transport any kind of file, email can be considered a textual document. Consequently, information retrieval oriented techniques have been applied to the management of email archives. Based on term-frequency descriptors (Baeza-Yates & Ribeiro-Neto (1999)), SOM-based analysis of Email archives has been proposed in Nuernberger & Detyniecki (2006). The authors trained a SOM for an email archive, and presented to the user the SOM grid on which selected keywords describing the content of Email documents represented by each SOM cell were printed. Supported by methods to search for keywords, the proposed system supports explorative analysis of Email archives.
A problem with Email may be the occurrence of large amounts of unsolicited Email ("spam") which detriments productivity of the Email users. In Keim et al. (2005), a SOM-based Email visualization system was proposed that leverages a color mapping scheme to visualize the degree to which SOM clusters contain spam email. Specifically, a spam classifier software was used to calculate for each email document its probability of being spam (so-called spam score, treated as a continuous attribute for each email). A color-mapping scheme was proposed which allows to simultaneously visualize the probability of each SOM cell to contain spam email, as well as the occurrence of selected keywords in the represented mails. To this end, for each SOM cell the spam scores of all represented Emails were averaged. That value was then mapped to a bipolar colormap going from blue (low scores, good mail) over white to red (high scores, spam mail). Thereby, red areas on the map indicate the presence of spam email, blue indicate the presence of valid email,while white areas indicate the border between both classes of email. Optionally, it was proposed to enhance the display by overlaying also the averaged frequency of user-selected keywords over each SOM cell. This was done by weighting the saturation channel of the mapped color by the normalized weight of the selected index term. Figure 2 illustrates a SOM trained of an archive of 10000 emails. Note that the visualization effectively combines two SOM views in one: A spam histogram (overall SOM structure) and a component plane (magnitude of a selected vector component).



(a) Spam histogram (b) Thresholded saturation (c) Scaled saturation

Fig. 2. SOM-based analysis of a spam/non-spam classified E-Mail dataset. (a) spam-histogram (red indicates spam, blue indicates non-spam). (b)-(c) represent the component plane for frequency of a selected term overlaid over the spam histogram by thresholding (b) or linearly scaling the spam histogram via the saturation channel of the respective colors.

4. Innovative Application Concepts

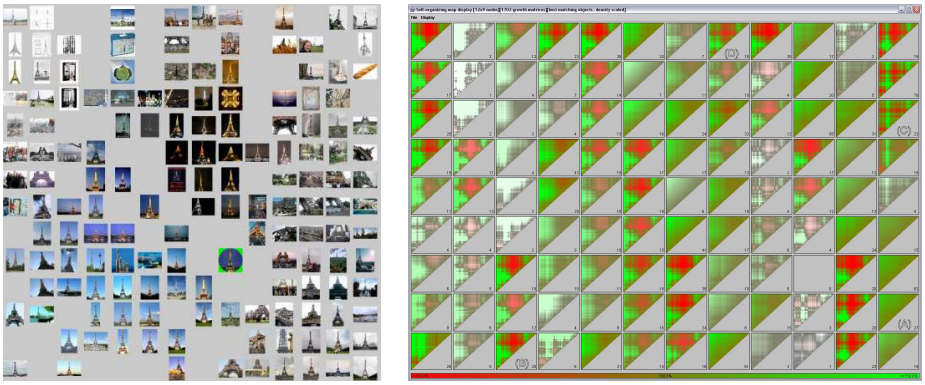
In this section, we review a number of recent SOM application concepts.

4.1 SOM-based Filtering for Image Retrieval

SOM to date has often been successfully applied in the Text (Nuernberger & Detyniecki (2006); Honkela et al. (1997)) and Multimedia Retrieval (Laaksonen et al. (2000); Pampalk et al. (2002); Bustos et al. (2004)) domains. Often, the goal is to provide an effective overview over the content of a document collection, to motivate the user to explore the collection (browsing), and to locate interesting documents which can in turn be used as query keys.
Image Sorter (Barthel (2008)) is an image search engine which shows particular deep integration of a SOM display of images with the core search engine. The overall goal is to provide efficient search in images on the Web by filtering the output of text-based Internet search engines. Specifically, Image Sorter uses Google image search to retrieve a seed set of images based on query by keyword. From the seed set, a SOM is trained using image color features. From the SOM-based overview, the user is then allowed to manually select a subset of the seed images (cf. Figure 3 for an illustration). These in turn are used in a subsequent retrieval stage. Specifically, a larger number of images is obtained using Google image search, yet this time the user selected image set is used to filter the larger answer set by content-based similarity with the selected image set. SOM training, image selection and retrieval are provided in real time, allowing effective image retrieval. Conceptually, the approach is regarded particularly appealing because it uses SOM to integrate an original content-based retrieval strategy with a keyword based web image search. The system is currently further developed within the Pixolution software suite (Pixolution. (2009)).

4.2 Small Multiple Views for Information Visualization

The sorting capabilities of the SOM algorithm make it applicable not only on general high-dimensional or multimedia data, but can also be used to sort sets of diagrammatic views or visualizations. In Information Visualization, often different views on the same data set are possible, and the user is confronted with selecting the most appropriate view. While one-by-one inspection of different views is a basic option, an overview of many views simultaneously can be helpful to quickly screen a large visualization space. As an example in this domain, in Schreck (2007) SOM was applied to sort many different views, effectively giving a SOM of many diagrams. The underlying diagrams are so-called Growth Matrices (Keim et al. (2006)), which encode in a color-coded triangular visualization all possible growth rates in a financial asset. The SOM was utilized to represent large sets of Growth Matrices in order to produce informative overviews. For training the SOM, the original growth matrix displays were down-sampled from matrices of edge lengths of several hundreds to matrices of size 20^2. The down-sampled matrices were used as Feature Vectors and made input to the SOM algorithm. A grid of 12×9 reference vectors was trained onto which the original data was mapped back. Figure 3 (b) illustrates a SOM obtained from the data described in Keim et al. (2006) by drawing the best matching Growth Matrix for each SOM node. Furthermore, a density histogram was overlaid over the display to show the frequency by which each Growth Matrix pattern is represented on the SOM. To this end, the saturation channel of each Growth Matrix thumbnail image was scaled proportional to the total number of objects matching the respective SOM cell.

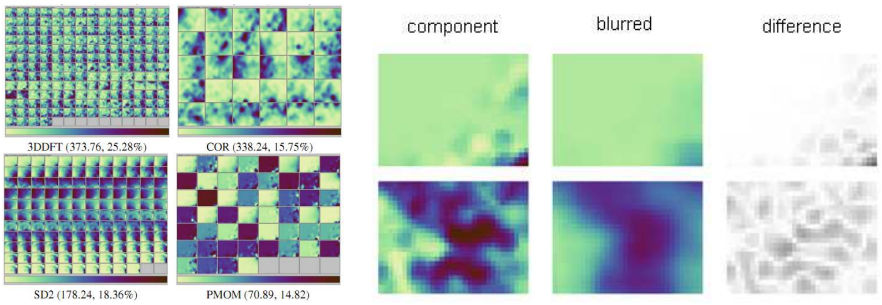


(a) Image Sorter system (b) Growth matrix display

Fig. 3. (a) The Image Sorter system uses color features to sort a set of images by similarity (Figure © 2008 IEEE, taken with permission from Barthel (2008)). The overview allows selection of query images for content based image retrieval. (b) Self-organizing map of 1.700 growth matrices. The SOM was trained from a down-sampled representation of the diagrams. Saturation scaling is performed on the thumbnail images to give additional information regarding the density distribution of the patterns.

4.3 Visual Feature Space Analysis

Complex data often needs to be described by Feature Vectors (FVs), before data analysis and similarity search tasks can take place. Often, many different FV representations of a given data set are possible, and the process of feature selection is expensive, often involving heuristics, human supervision, and benchmarking. In Schreck et al. (2006), Umatrix visualizations of a given data set described by different FV representations were compared. It was observed that Umatrices of different FV representations differed significantly regarding the distribution of distances between adjacent reference vectors (shown as gray- or color-codings in Umatrix displays). Regression experiments found a correlation between the uniformity of the distance distributions, and the discrimination capability of the underlying FV representations, as measured by ground truth benchmark information. The hypothesis was formulated that discriminative feature spaces can be expected to provide a rich mix of distances between adjacent reference vectors (forming high contrast images), while non-discriminating feature spaces tend to show more skewed Umatrix distance distributions. That hypothesis was further evaluated by experiments on synthetic data (Schreck, Fellner & Keim (2008)) as well as also considering distribution characteristics observed in component plane images (Schreck, Schneidewind & Keim (2008)). Figure 4 illustrates the latter analysis for several different feature representations of a benchmark data set. While more experimental and theoretical consideration is needed, the idea of applying unsupervised SOM analysis to support the feature selection process is considered promising.

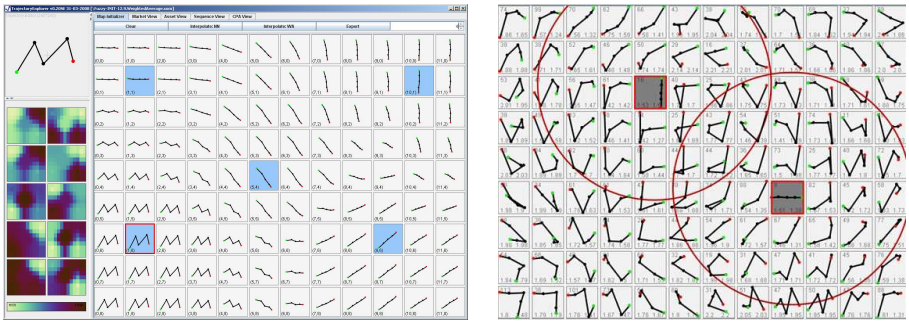


(a) Component plane arrays (b) Analysis of component plane images

Fig. 4. (a) Arrays of component planes for four different feature vector space representations of the same benchmark data set. (b) A *Difference-of-Gaussians* analysis yields a measure for the amount of information contained in a component plane image. An overall measure for each feature space is obtained by averaging over the Difference-of-Gaussian measures of each component plane image in that feature space. That score in turn can be used as a heuristic criterion for unsupervised feature selection, or to filter large sets of candidate feature spaces for analysis by an expert user.

4.4 Interactive Trajectory Analysis

Besides multimedia and general multidimensional data, SOM-based visual analysis of time-dependent data has also been successfully done in the past. A system considering SOM-clustering of one-dimensional time series data is described in Šimunić (2003). Two-dimensional time series in form of trajectories derived from scatter plot data have been analyzed with SOM in Schreck et al. (2007). Most SOM-based analysis systems operate by training the SOM in off line mode, and restrict to static visualization of the SOM analysis output. In Schreck et al. (2009), a concept for fully visual-interactive training of SOMs for trajectory pattern data is introduced. The basic idea is not only to visualize the SOM result as is evolves during training in real time, but also, to provide the user means to initialize and control the SOM algorithm interactively. To this end, an editor allows the user to visually initialize the grid of trajectories, and adjust important training parameters. At any time during the training, the user is able to pause the training, update specific parameters as is seen fit, and then continue or restart the training. Figure 5 (a) shows the interactive initialization of the prototype vector array, which is possible due to the specific vector representation of the trajectory data (Schreck et al. (2009)). Figure 5 (b) illustrates interactive control of the training process. In the example, the user manually adjusts two reference trajectories, and also updates the neighborhood kernel for the next training iterations to take place.



(a) Initialization (b) Interactive Control of Training

Fig. 5. (a) Interactive initialization of a grid of prototype trajectories prior to start of the SOM training. (b) Visual-interactive update of selected reference vectors and training parameters during online training.

5. Selected Research Challenges

In the previous sections, we have illustrated a number of examples which in our view represent innovative SOM-based visualization approaches and application concepts. For both aspects, we summarize the findings from the presented examples, and provide a selection of future work we consider promising to undertake.

5.1 SOM Visualization

Effective SOM-based data analysis relies on appropriate visualization of the output of the SOM algorithm. The visualization approaches presented in this article enhance standard SOM displays by (i) integrating structural properties of prototype components into the Umatrix display (Section 3.1); (ii) by using vector fields for improved visual cluster analysis (Section 3.2); and (iii) by mapping not one but two quantitative variables to color in a distribution view (Section 3.3). We observe that as the SOM algorithm provides rich information, the provision of displays adequately representing this information is important to allow effective interactive SOM analysis. The presented approaches aim to increase the density of the information visualized. Inspired by this review, interesting future work in SOM visualization may be identified along the following lines:

5.1.1 Visualization of High-Dimensional Data

The network of SOM prototype vectors and its associated sample distribution can be regarded as a high-dimensional visualization problem. Much potential is perceived in appropriately combining standard solutions from high-dimensional data visualization with the SOM data structure. E.g., appropriately designed glyph- or parallel coordinate plots, reflecting also the neighboring relationships, could help in packing more information into a single static SOM view.

5.1.2 Comparative SOM Visualization

Often, the data under concern can be described by many different feature representations, each reflecting different similarity notions. As it often is not clear which of these representations is the best suited, it is proposed to develop specific visualizations for comparative SOM

analysis. A connector-oriented approach as introduced in Holten (2006) could be suited as an appropriate tool to quickly compare the outcome of training several SOMs for the same data set described under different vector representations. Also on the interaction side, comparison of different SOMs of the same data set could be supported by e.g., means of animation. Then, how to transit between different SOMs is an interesting problem to solve.

5.1.3 Visualization of Sample Distribution Characteristics

A key SOM-based analysis task refers to analysis of the distribution of data samples over the SOM. In many cases, the number of data samples outperforms the number of SOM cells. For effective distribution visualization, appropriate visual representations of sets of samples need to be found, for overlay over the display. For many data types such as textual or multimedia documents, finding the right representations (e.g., labels or appropriately defined thumbnail previews) is a difficult problem, the solution of which could further contribute to advanced SOM visualization.

5.1.4 Visualization of SOM Training Process

Currently, in most cases only the final output of the SOM training, usually done in a black-box manner, is the basis for visualization. However, the online visualization of the training process is expected to help the user better understand emerging data structure yielded by the SOM. Also, visualization of the training could help in finding better parameterizations by the user, by means of interaction with the training algorithm itself. While one system that visualizes the SOM training on a specific data type (trajectories) has been presented in Section 4.4, the visualization of SOM training for general data types is an interesting problem for future work.

5.2 Application Concepts

On the application side, we have reviewed a selection of SOM-based application concepts that (i) showed the deep application integration of SOM analysis in a retrieval system (Section 4.1); (ii) SOM as a layout generator for diagram visualization (Section 4.2); (iii) visual feature space analysis (Section 4.3); and (iv) interactive SOM training (Section 4.4). These are only a few of many new SOM applications introduced which demonstrate that the potential for SOM applications is far from being exhausted. A number of interesting future research directions can be outlined in the application context as follows.

5.2.1 SOM quality assessment

The quality of a given SOM can be measured by many aspects, ranging from analytic measures of quantization error or topology preservation to user-oriented measures based on interpretability, subjective notions of data similarity, etc. The definition of a set of flexible quality assessment capabilities would benefit many application areas, in leading to better quality assessments by the user. Also, SOM quality assessment should be tightly coupled with interactive parameterization of the SOM training process, to allow the user to instantly compare and re-generate different SOM results, to obtain results best suiting the given application and data set.

5.2.2 SOM Postprocessing and Analysis Support

SOM analysis often includes considerable user effort during an extensive interpretation stage. Many post processing techniques can in principle support the interpretation stage. Examples include post processing for automatic detection of SOM clusters, or comparison of the SOM

output with those of other clustering algorithms. Integration of such analysis into the SOM application could increase effectiveness of SOM analysis by the user.

5.2.3 Applications in Information Retrieval

Browsing and searching in data are closely related. In multimedia retrieval, query-by-example is a popular query concept, in which the user supplies a multimedia object as a query key to search for. Obtaining appropriate query examples requires efficient browsing capabilities, allowing the user to quickly screen large data sets. While the SOM is well suited to provide the basic structure for browsing, issues relating to the presentation of sets of elements and the efficient selection of individual elements thereof is a non-trivial problem. We see a need to develop further methods to more tightly integrate SOM-based browsing facilities with query-by-example oriented retrieval.

6. Conclusions

We surveyed a set of SOM-based works illustrating, as we consider, innovative visualization approaches and application concepts. We argued that while the base SOM algorithm is already well-known and widely used, much innovation potential exists in further improving SOM visualization and defining novel application areas. Specifically, the Information Visualization field offers a rich background of visualization approaches which can be exploited for advanced SOM visualization. Deeper application integration and visual-interactive control of the SOM training process are regarded promising for future applications. The tight integration of automatic data analysis and visualization, as demonstrated by many SOM-based systems, offers much potential in handling increasing amounts of complex data, and is considered a promising basis for efficient information retrieval systems.

Acknowledgments

We thank Tatiana von Landesberger, Juergen Bernard, Sebastian Bremm, and Joern Kohlhammer for fruitful discussion of SOM-based visual analysis methods. Figures 1 (a+b) are courtesy of Andreas Rauber. Figure 3 (a) is courtesy of Kai Uwe Barthel.

7. References

Baeza-Yates, R. & Ribeiro-Neto, B. (1999). *Modern Information Retrieval*, Addison-Wesley.

Barthel, K. U. (2008). Improved image retrieval using automatic image sorting and semi-automatic generation of image semantics, *Image Analysis for Multimedia Interactive Services, International Workshop on* **0**: 227–230.

Bustos, B., Keim, D., Panse, C. & Schreck, T. (2004). 2D maps for visual analysis and retrieval in large multi-feature 3D model databases, *Proceedings of the IEEE Visualization Conference (VIS'2004)*, IEEE Press. Poster paper.

Card, S., Mackinlay, J. & Shneiderman, B. (1999). *Readings in Information Visualization: Using Vision to Think*, Morgan Kauffman.

Duda, R., Hart, P. & Stork, D. (2001). *Pattern Classification*, 2nd edn, Wiley-Interscience, New York.

Guo, D., Chen, J., MacEachren, A. M. & Liao, K. (2006). A visualization system for space-time and multivariate patterns (VIS-STAMP), *IEEE Transactions on Visualization and Computer Graphics* **12**(6): 1461–1474.

Hinneburg, A., Wawryniuk, M. & Keim, D. A. (1999). HD-eye: Visual mining of high-dimensional data, *IEEE Computer Graphics & Applications Journal* **19**(5): 22–31.

Holten, D. (2006). Hierarchical edge bundles: Visualization of adjacency relations in hierarchical data, *IEEE Transactions on Visualization and Computer Graphics* **12**(5): 741–748.

Honkela, T., Kaski, S., Lagus, K. & Kohonen, T. (1997). WEBSOM—self-organizing maps of document collections, *Proceedings of WSOM'97, Workshop on Self-Organizing Maps, Espoo, Finland, June 4-6*, Helsinki University of Technology, Neural Networks Research Centre, Espoo, Finland, pp. 310–315.

Inselberg, A. & Dimsdale, B. (1990). Parallel coordinates: a tool for visualizing multi-dimensional geometry, *VIS '90: Proceedings of the 1st conference on Visualization '90*, IEEE Computer Society Press, Los Alamitos, CA, USA, pp. 361–378.

Keim, D. A., Ankerst, M. & Kriegel, H.-P. (1995). Recursive pattern: A technique for visualizing very large amounts of data, *VIS '95: Proceedings of the 6th conference on Visualization '95*, IEEE Computer Society, Washington, DC, USA, p. 279.

Keim, D., Mansmann, F. & Schreck, T. (2005). Mailsom - visual exploration of electronic mail archives using Self-Organizing Maps, *Second Conference on Email and Anti-Spam (CEAS 2005), Stanford University, Palo Alto, CA, USA, July 21-22*. Short paper.

Keim, D., Nietzschmann, T., Schelwies, N., Schneidewind, J., Schreck, T. & Ziegler, H. (2006). A spectral visualization system for analyzing financial time series data, *Proceedings of the EuroVis 2006: Eurographics/IEEE-VGTC Symposium on Visualization, Lisbon, Portugal, May 8-10 , 2006*, IEEE Computer Society.

Kohonen, T. (2001). *Self-Organizing Maps*, 3rd edn, Springer, Berlin.

Laaksonen, J., Koskela, M., Laakso, S. & Oja, E. (2000). PicSOM—content-based image retrieval with self-organizing maps, *Pattern Recogn. Lett.* **21**(13-14): 1199–1207.

Liu, H. & Motoda, H. (eds) (2007). *Computational Methods of Feature Selection*, Data Mining and Knowledge Discovery, Chapman & Hall/CRC.

Neumayer, R., Mayer, R., Poelzlbauer, G. & Rauber, A. (2007). The metro visualisation of component planes for self-organising maps, *Proceedings of International Joint Conference on Neural Networks*, IEEE.

Nuernberger, A. & Detyniecki, M. (2006). Externally growing self-organizing maps and its application to e-mail database visualization and exploration, *Applied Soft Computing* **6**(4): 357–371.

Pampalk, E., Rauber, A. & Merkl, D. (2002). Using smoothed data histograms for cluster visualization in self-organizing maps, *Proc. Int. Conf. on Artifical Neural Networks (ICANN'02)*, Vol. 2415 of *Lecture Notes in Computer Science*, Springer.

Pixolution. (2009). Visual image search software. http://www.pixolution.de/.

Polzlbauer, G., Dittenbach, M. & Rauber, A. (2005). A visualization technique for self-organizing maps with vector fields to obtain the cluster structure at desired levels of detail, *Neural Networks, 2005. IJCNN '05. Proceedings. 2005 IEEE International Joint Conference on*, Vol. 3, pp. 1558–1563 vol. 3.

Schreck, T. (2007). *Effective Retrieval and Visual Analysis in Multimedia Databases*, PhD thesis, University of Konstanz, Germany.

Schreck, T., Bernard, J., von Landesberger, T. & Kohlhammer, J. (2009). Visual cluster analysis of trajectory data with interactive kohonen maps, *Information Visualization* **8**(1): 14–29.

Schreck, T., Fellner, D. & Keim, D. (2008). Towards automatic feature vector optimization for multimedia applications, *SAC '08: Proceedings of the 2008 ACM symposium on Applied computing*, ACM, New York, NY, USA, pp. 1197–1201.

7

Tracking and Visualization of Cluster Dynamics by Sequence-based SOM

Ken-ichi Fukui[1], Kazumi Saito[2], Masahiro Kimura[3] and Masayuki Numao[1]
[1]Osaka University, [2]University of Shizuoka, [3]Ryukoku University
Japan

1. Introduction

Since events and physical phenomena change with time, it is important to capture the main transitions and elements of such events and phenomena. Such transitions can be seen to occur in the World Wide Web (Levene & Poulovassilis, 2004), news topics (Allan, 2002), a person's health condition, and the state of an instrument or a plant. Transition, or change, refers to a sequential increase/decrease or generation/extinction of the feature of the object. Visualization of such transitions of dynamic clusters is helpful in understanding such phenomena instinctively and plays a useful role in many application domains, such as fault diagnosis and medial examinations.

Although a number of clustering methods have been proposed (Jain et al., 1999), most conventional clustering methods deal with static data and cannot handle sequential changes of the cluster explicitly. Tracing the trajectory within clusters that have been collectively processed and a sliding window-based method to generate separate clusters can be considered as simple methods. Although the former method cannot trace changes of clusters, it can trace changes in the number of data that belong to each cluster. The latter method can handle changes of clusters to a certain degree. However, there are some problems, such as setting an appropriate window size, the inevitable decrease in the number of data within a window, and the correspondence relationships of clusters between windows.

The present study considers a window-based approach using the temporal neighborhood to address the above-described problems. Kohonen's Self-Organizing Map (SOM) (Kohonen, 2000) is considered to be an appropriate technique for visualizing clusters and their similarity relationships. The SOM is an unsupervised neural network learning technique that produces clusters and subsequently projects them onto a low-dimensional (normally two-dimensional) topology map. The conventional SOM deals with static data. However, we have extended the SOM learning model by introducing the Sequencing Weight Function (SWF), so that the model can visualize the transition of dynamics clusters. This model is referred to herein as the Sequence-based SOM (SbSOM) (Fukui et al., 2008). A SOM-based method was selected because the SOM has a neuron topology in the feature space and that is associated with topology (visualization) space. The introduction of temporal order into the topology is natural. The proposed method mitigates the problems of appropriate

window size and the decrease of the number of data. Moreover, owing to the neighborhood function, the spatio-temporal neighborhood become the topological neighborhood, the correspondence relation problem of clusters can be solved.
Projection methods include conventional Multi-Dimensional Scaling (MDS), using the first and second components of Principal Component Analysis (PCA), and Locally Linear Embedding (LLE) (Roweis & Saul, 2000), and graph-based methods include spring embedding, the Laplacian eigenmap (Belkin & Niyogi, 2002), and the ISOMAP (Tenenbaum et al., 2000). However, these methods project each sample, because clustering is not performed. As such, these are not suitable for large data sets.
A number of studies have been conducted to track cluster changes, but these studies have concentrated on tracking topic changes. For example, TimeMines (Swan & Jensen, 2000) extracts topics and visualizes the periods and intensities of the topics by a χ^2 test from a sequence of words related to the documents. T-Scroll (Ishikawa & Hasegawa, 2007) proposed the introduction of a temporally gradual decrease model to K-means clustering. However, this does not clearly solve the problem of correspondence of clusters between windows. As extension of the static topic model to a dynamic model, Kimura et al. (2005) proposed the multinominal PCA topic model to extract latent topic trends, and Wang and McCallum (2006) proposed the Latent Dirichlet Allocation (LDA) based topic model over time. In addition, in network analysis, Qian et al. (2009) extracted influential research topics and tracked them using a network of research paper citations using the community inference method. An advantage of the proposed approach compared to previous studies is that the SbSOM is a general framework so that it is applicable to any data that can be represented by vectors. In addition, the SbSOM is easy to extend to graph, string, and other structured data by applying the recently developed kernel method (Bishop, 2006). Moreover, the kernelized SOM has been proposed (Lau et al., 2006).
On the other hand, various approaches have been proposed to introduce the concept of time to the SOM model (Barreto & Arajo, 2001). However, these approaches have been introduced as physiological models for short-/long-term memory, or to handle time series data. In contrast, the purpose of the present study is to track cluster changes.
We applied the proposed method to real-world applications, a news articles data set (Fukui et al., 2008), a medical data set (Fukui et al., 2006), and an Acoustic Emission (AE) signal data set (Fukui et al., 2007). Based on the results of these applications, we have confirmed that the map obtained using the SbSOM can interpret as a phenomenon, topic transition from a news articles data set and cluster transition of patients from a medical data set.
The remainder of the present chapter is organized as follows. Section 2 describes the characteristics of cluster dynamics. Section 3 introduces the learning algorithm of the proposed SbSOM. Section 4 presents an application of SbSOM to transition of news topics. Section 5 introduces a visualization method using class labels for sequence data. Section 6 presents an application involving medical data. Finally, conclusions are presented in Section 7.

2. What is Cluster Dynamics?

This section describes which characteristics of cluster change should be tracked. Cluster change can be represented by the following basic three characteristics (see also Fig. 1):

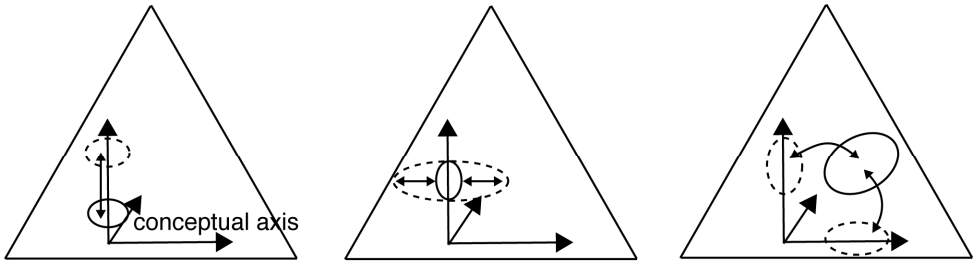


Fig. 1. Characteristics of cluster dynamics. These figures illustrate high-dimensional space based on the concept vector, e.g., a topic represented by a certain direction in the Bag-of-Words vector space.

1) **Movement:** a change in the feature of the cluster. In particular, if a certain direction in the feature space represents a certain concept, e.g., topic, movement and direction indicate the increase/decrease of the degree of the concept, e.g., the level of interest in a topic.
2) **Range:** a change in coverage representing the variety of the cluster. For instance, the divergence/convergence of a topic.
3) **Merger/Separation:** a change wherein more than two clusters merge into one cluster or one cluster separates into multiple clusters. This can be interpreted as a change in topic, i.e., the derivation of a topic. (However, mergers of topics appear to be rare.)

3. Sequence-based Self-Organizing Map

3.1 Algorithm

The Sequence-based SOM (SbSOM) is based on Kohonen's Self-Organizing Map (SOM). Let v-dimensional N inputs be $X_n = (x_{n,1}, \cdots, x_{n,v})$, $(n = 1, \cdots, N)$. Let the position of M neurons in the visualization layer be $r_m = (y_m, z_m)$, $(m = 1, \cdots, M)$, and let the reference vector corresponding to the m^{th} neuron be $W_m = (w_{m,1}, \cdots, w_{m,v})$.

The following is the learning algorithm that uses a batch process and the decreasing strategy of the learning parameter.

Batch sequence-based SOM learning algorithm

Step 1. Initialize the reference vectors $\{W_1, \cdots, W_M\}$ randomly.

Step 2. Search winner neurons for all inputs $\{X_1, \cdots, X_N\}$. (This step is described in the next subsection.)

Step 3. Exit if the winner neurons $\{c(X_1), \cdots, c(X_N)\}$ were not changed.

Step 4. Update the reference vectors $\{W_1, \cdots, W_M\}$ by the following equation:

$$W_m^{new} := W_m + h_{c(X_n),m}[X_n - W_m] \quad (1)$$

where $h_{c(X_n),m}$ is a neighborhood function that defines the effect of the neighborhood of the winner. Typically, a Gaussian function is used:

$$h_{c(X_n),m} = \alpha \exp\left\{-\frac{\left\|r_m - r_{c(X_n)}\right\|}{2\sigma^2}\right\} \tag{2}$$

Step 5. Decrease the learning parameters σ every several iterations. Return to Step 2.

3.2 Sequencing Weight Function

In the conventional SOM, the winner neuron is determined only by spatio-distance. In contrast, in the SbSOM, by introducing the Sequencing Weight Function (SWF), weights are assigned to the neuron topology according to the sequence of the data (Fig. 2). The SWF introduces the concept of time to the topology. Note that this is a loose concept of time because it allows reversal of the data order that appears on the neuron topology by the winner. The winner is determined by spatio-temporal distance using SWF $\phi(n,m)$ as follows:

$$c(X_n) = \arg\min_m \phi(n,m)\left\|X_n - W_m\right\|. \tag{3}$$

Suppose t_n is a time stamp for the n^{th} data, and that the m^{th} neuron is located at η_m / η_M in a certain direction on the topology (in this case, the η-direction). Let the absolute value of the difference in these ratios be $\varepsilon = \left|t_n / t_{end} - \eta_m / \eta_M\right|$. When there is no time stamp associated with the data, but the data is obtained sequentially, ε is given as $\varepsilon = \left|n/N - \eta_m / \eta_M\right|$. Then, the SWF is defined so as to be able to balance the spatio-temporal resolution by allowing reversal of data order:

$$\phi_{\exp}(n,m) = e^{\beta\varepsilon}, \tag{4}$$

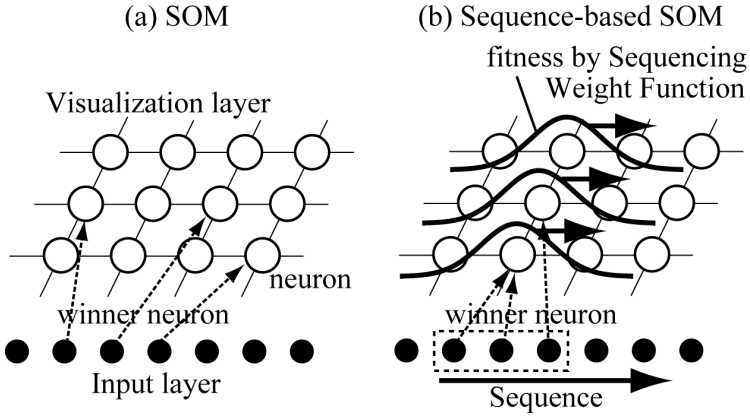


Fig. 2. Difference in winner neuron selection. (a) Determined only by spatio-distance in the SOM. (b) Determined by spatio-distance under the sequencing weight function in the SbSOM.

where $\beta > 0$ is a parameter that controls the degree of influence of the data order, i.e., temporal distance. As β increases, the temporal distance becomes increasingly dominant. Instinctively, by shifting the SWF in a certain direction on the neuron topology, the SbSOM can produce time axis onto the topology (Fig. 2). Note that when $\beta = 0$ (i.e., $\phi(n,m) = 1$), SbSOM will be exactly equivalent to the standard SOM.
In case of style of restricted sliding window, the SWF can be given by a rectangle function:

$$\phi_{rect}(n,m) = \begin{cases} 1 & \varepsilon < 1/2K \\ \infty & otherwise \end{cases} \tag{5}$$

where K is the number of neurons in a certain direction (e.g., the η-direction). When β in is taken to be sufficiently large, $\phi_{\exp}$ is equivalent to ϕ_{rect}. The advantage of the sliding window is reduced computational cost, because the window requires only comparison with reference vectors within a window.

3.3 Neighborhood Function in the SbSOM

The SOM includes the neighborhood function, so that the reference vectors are updated according to the spatio-neighborhood. The neighborhood function in the SbSOM, on the other hand, is a spatio-temporal neighborhood related to SWF (Fig. 3). Therefore, the SbSOM can perform clustering with the help of temporal neighborhood data even when using ϕ_{rect}.

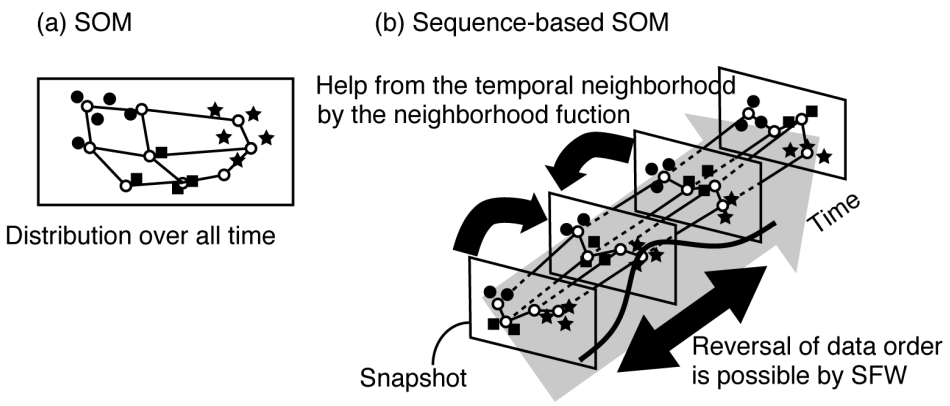


Fig. 3. The spatio-neighborhood becomes the topological neighborhood (a) in the SOM and (b) in the SbSOM. This figure shows a restricted version of the SbSOM.

This property mitigates the problems of appropriate window size and decreased sample data in window-based clustering. Moreover, cluster correspondence can be self-organized because, in the SbSOM, the spatio-temporal neighborhood becomes the topological neighborhood.

4. Experiment 1: Transition of News Topics

4.1 Dataset and Pre-processing

This experiment uses international articles from the 'Mainichi' Japanese newspaper from January to December of 1993[1]. The total number of articles is 5,824, and the total number of unique words is 24,661, after eliminating pre-defined stop words. Each article is represented by a Bag-of-Words model, namely a term frequency - inverse document frequency (tf-idf) model.

Assuming a topic is distributed around certain direction in the vector space of the words, we applied Principal Component Analysis (PCA) to extract topic axes (feature axes). The dimension of the weight vector of the words is reduced by projecting the words into the topic axes space. In addition, a topic class (label) is assigned to each article by the topic axis index with the maximum component. See (Kimura et al., 2005) for a detailed description of topic extraction. Note that the SbSOM is applicable to any topic extraction method as long as the topics are represented by vectors.

In this experiment, the number of PCs was set to 10, which was empirically adjusted. Therefore, the number of extracted topics was 20 as positive and negative direction of each PC indicates different topic, since the origin is the center of the entire articles. The number of articles associated with the topics and the topic titles are listed in Table 2. These topic titles were assigned by two individuals who read articles associated with the topics and merged the topics to obtain consistent titles.

4.2 Results for the News Article Dataset

Visualization results using the conventional SOM and the SbSOM are shown in Fig. 4 and Fig. 5, respectively. The numbers denoted in the figures indicate the representative topic of the cluster as determined by majority decision of topic labels of the articles belonging to the cluster. The neuron topology was set to 24x20 (regular grid), and $\phi_{\exp}$ was used for the SWF with the parameter $\beta = 150.0$. In the SbSOM (Fig. 5), the x direction is the time axis, as indicated by the months of the year.

Clearly, the SOM can perform clustering of related articles (Fig. 4), whereas the SbSOM also provides change of topics. In addition, Fig. 6 shows the intensity distribution of each component that represents a topic, which indicates the level of interest in a topic. For example, the level of interest in the 7th topic (Chinese Situation) is high around March and December, reflecting the election of Chinese president on March 27th and the 100th anniversary of Mao Tse-Tung's birth, respectively.

No.	# of articles	Topic – Subtopics
1	453	Russian situation - Opposing President Yeltsin and the national assembly - Creating a new constitution
2	32	--
3	303	Middle East peace problem - Israel/PLO agreement and trends of involved nations
4	43	--

[1] http://www.nichigai.co.jp/sales/mainichi/mainichi-data.html (in Japanese)

5	471	Cambodian general election
6	148	North Korea matter - Nuclear problem
7	721	Chinese situation - Economics - Change of government
8	452	Bosnian conflict - Bosnian peace conference - Tendency of U.S.A. and the United Nations
9	171	Military and diplomacy in U.S.A. - Foreign policy of Clinton administration
10	290	Bosnian conflict - Muslim influence - Movement of Muslims
11	366	Iraq problem
12	436	North Korea matter - Nuclear problem - North and South Korea conflict
13	345	Japan and foreign countries - Postwar compensation issue - Increasing role in the United Nations
14	230	Middle East situation - General election in Pakistan
15	327	North Korea and China
16	260	EC Integration Summits - including APEC, EC, ASEAN…
17	253	Asian Situation - including China, Taiwan, Korea…
18	128	Russian circumstance - Election of a new national assembly
19	127	Elections in various countries
20	268	Miscellaneous conferences

Table 2. Extracted topics and manually assigned titles.

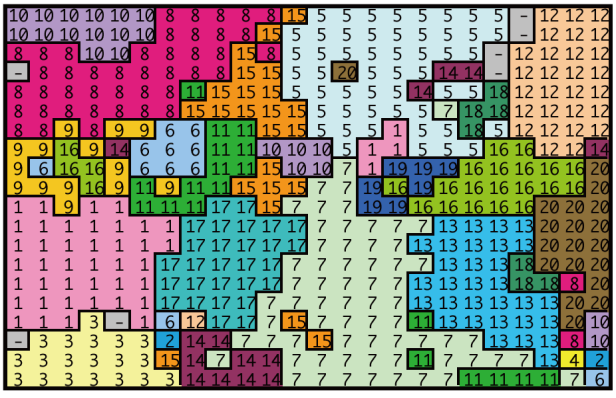


Fig. 4. Mapping result by SOM (represented by majority topic).

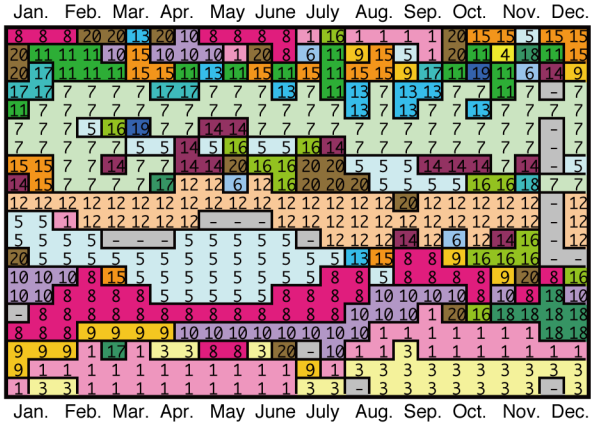


Fig. 5. Mapping result by Sequence-based SOM (represented by majority topic).

Derivation of a Topic

The 9th topic is related to military affairs and diplomacy in the United States. The principal event was the inauguration of President Clinton on January 20th. Therefore, the level of interest in the 9th topic was high around January (Fig. 6). The level of interest in the 9th topic was also high from around March to June, where the majority topic of these clusters is the 5th topic in Fig. 5, which means these two topics are highly related each other. In addition, the levels of interest in the 6th and 11th topics were high around January, where the majority topic is the 9th topic. The 5th topic (Cambodian General Election), the 6th topic (North Korean Nuclear Problem), and the 11th topic (Iraq Problem) are topics that are closely related to President Clinton. That is to say the map shows engagement of President Clinton to these topics.

Furthermore, the 8th and 10th topics are both related to the Bosnian conflict. However, the sub-topics are different for these topics. These topics appear close to each other within the overall map (Fig. 5). In addition, Fig. 6 shows that the 10th topic, which is related to the movement of Muslims, is derived from the 8th topic.

Diversification and Convergence of a Topic

Diversification/convergence of a topic is represented by expansion/contraction in the y direction with the map. Since the y direction in this SbSOM indicates that the neighborhood on the Bag-of-Words vector space is the same as that in the conventional SOM, expansion in y direction represents an increase of the variety of words related to the topic, i.e., diversification of the topic. For example, the variety of words related to topic #5 (Cambodian General Election) increases toward the general election in May and then decreases. Diversification and convergence can also be seen in the component maps (Fig. 6).

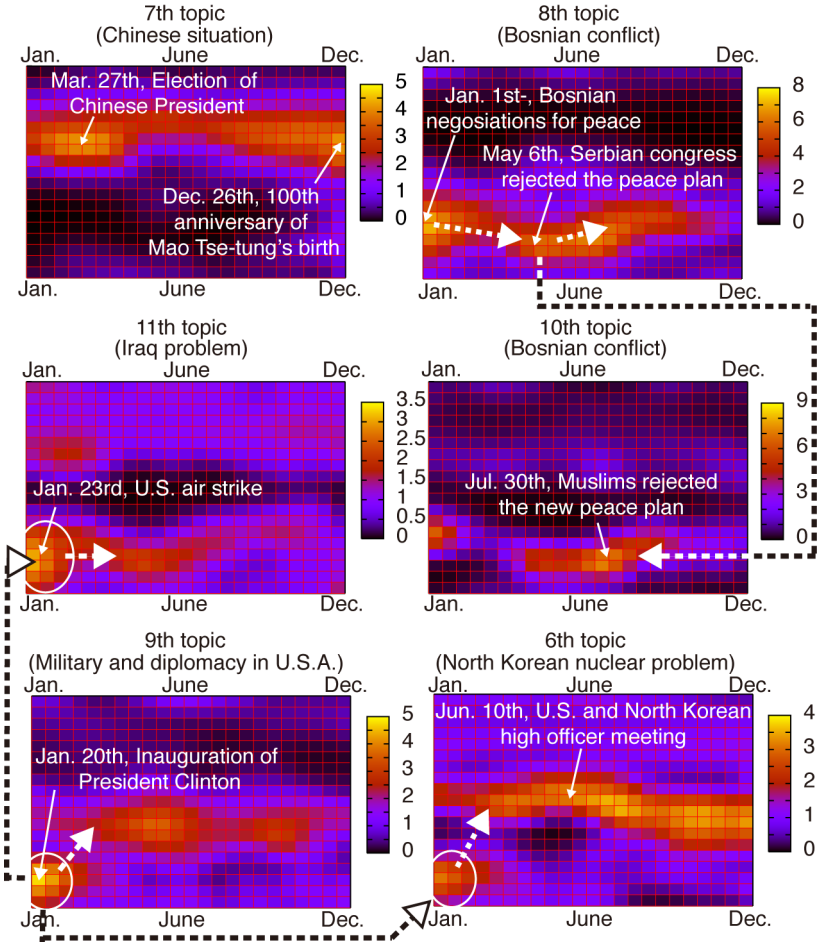


Fig. 6. Maps of level of interest in each topic corresponding to Fig. 5. Overlapped region of high level in different topics indicate relation between the topics. Considering time ordering of high level regions, dotted lines indicate derivation of the topic.

4.3 Effect of the Sequencing Parameter

In this subsection, we investigate the effect of the sequencing parameter in the SWF on visualization. Figure 7 shows one component changing the parameter β. When $\beta = 0$, the SbSOM is exactly the same as the conventional SOM. As β increases, the level of interest gradually spreads in the time direction and stabilizes after $\beta = 100$. This parameter is not sensitive to the result if it is sufficiently large value.

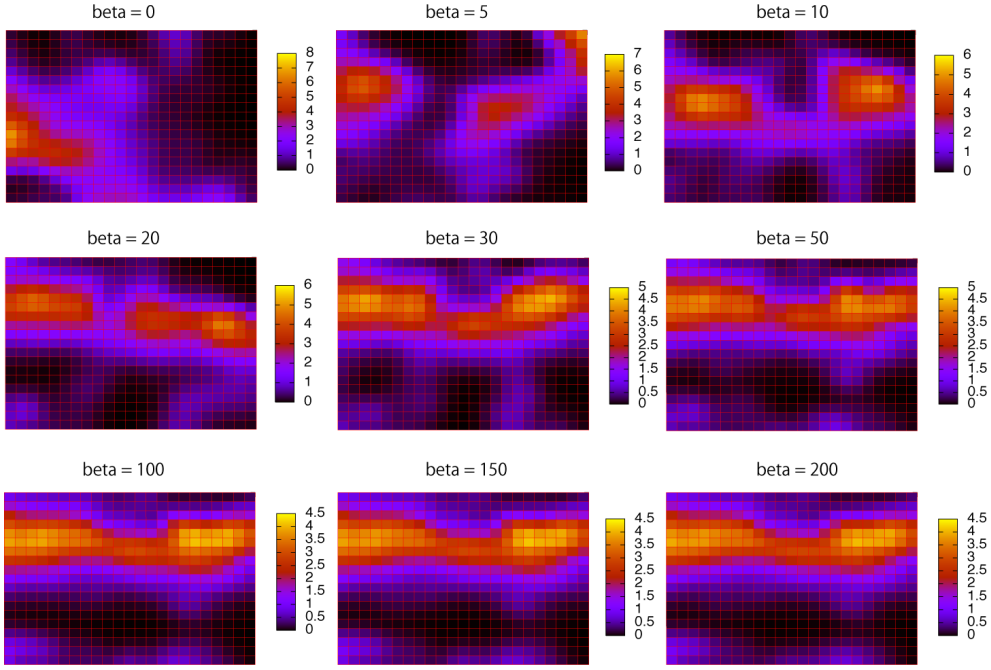


Fig. 7. Effect of the sequencing parameter on visualization.

5. Application to Two Classes of Sequential Data

5.1 Concept

This section presents the visualization architecture for sequential data using the class label. The conventional SOM does not consider the class label while learning because the learning in the SOM is an unsupervised learning. The nature of unsupervised learning is to find latent clusters within data or to find relationships between attributes. However, it is natural to use class label information when available. Although a few studies have extended the SOM to the supervised SOM, e.g., Fritzke, 1994, Hagenbuchner & Tsoi, 2004, these studies did not preserve the nature of the unsupervised learning.

Consequently, preserving the nature of unsupervised learning, we proposed the idea of using the class label to highlight clusters that are related to a specific class (Fig. 8). After SbSOM learning, trajectories of the winner neuron indicating a series of data are obtained. Afterwards, weights assigned to neuron nodes are trained by a single-layer perceptron. The obtained weights are used for visualization, e.g., color gradation. A period that has high correlation with a class is expected to have high weights, and vice versa. The reason for selecting a single-layer perceptron is so that connectivity weights can be paired with neuron nodes.

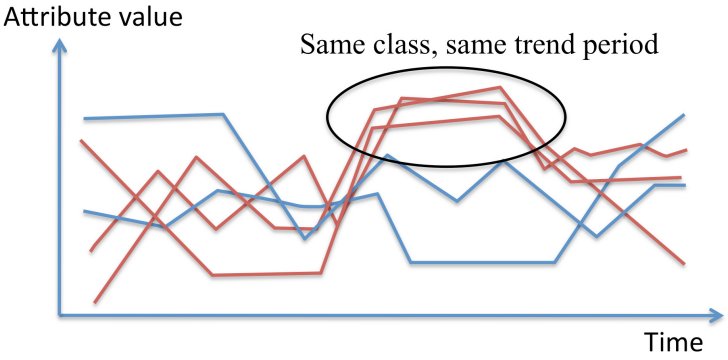


Fig. 8. Concept of using the class label for sequence data. Periods having the same class and trend should be highlighted.

5.2 The algorithm

The proposed visualization architecture is illustrated in Fig. 9. The advantages of this methodology are to derive the interpretation of the discriminant function for the map via the perceptron and to suggest the cluster in a sequence that involves classification through the SOM.

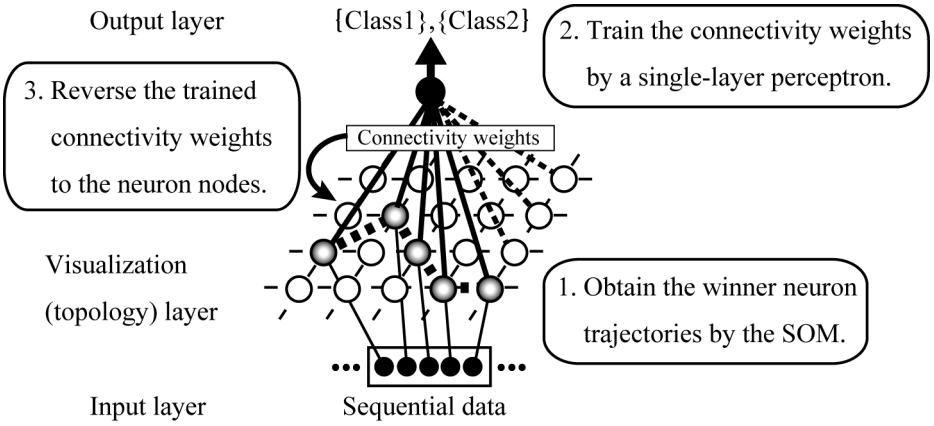


Fig. 9. Visualization architecture for two classes of sequential data using the class label.

Let k be an index of K sequential data, and let n_k be the number of elements in the k^{th} sequential data. The k^{th} sequential data is represented by $\left\{\left(x_r^{(k)}, t_r^{(k)}\right): r = 1, \cdots n_k\right\}$, where $\sum_{k=1}^{K} n_k = N$. The visualization architecture consists of three steps:

Steps of learning weights for visualization

Step 1. Train the reference vectors using the Sequence-based SOM with $\left\{\left(x_r^{(k)}, t_r^{(k)}\right): r = 1, \cdots n_k, k = 1, \cdots, K\right\}$ as input data. For instance, the first sequence of the first datum $\left(x_1^{(1)}, t_1^{(1)}\right)$ is an input. Consequently, the winner neuron trajectories are obtained within the visualization layer.

Step 2. Construct a discriminant function by a single-layer perceptron using the class label as the winner neuron trajectories are input. Concretely, let an input vector be $Y_k = (y_{k,1}, \cdots, y_{k,M})$. Set $y_{k,m}$ corresponding to the m^{th} neuron node for the k^{th} sequence as follows:

$$y_{k,m} = \begin{cases} 1/n_k & if\ \exists r\, X_r^{(k)} \in C_m \\ 0 & otherwise \end{cases} \tag{6}$$

Let the connectivity weights be $W = (w_1, \cdots, w_M)$. The discriminant function is represented by $\hat{z}_k = W \cdot Y_k$. Let $z_k \in \{-1,1\}$ be class labels. Obtain the optimal weights by minimizing the following objective function:

$$E = \sum_{k=1}^{K} \left(z_k - \hat{z}_k\right)^2. \tag{7}$$

When the objective function is minimized by the gradient decent method, connectivity weights W are updated by the following equation:

$$w_m^* \coloneqq w_m + \gamma \sum_{k=1}^{K} \left(z_k - \hat{z}_k\right) y_{k,m}\,, \tag{8}$$

where γ is the learning rate.

Step 3. Finally, the map is visualized by reversing the obtained weights into the neuron nodes and setting the weights as contracting density.

6. Experiment 2: Time Series of a Medical Dataset

6.1 Dataset and Problem Setting

We used time series of blood test data of 137 hepatitis C patients for one year collected from a Japanese hospital (the medical faculty of Chiba University). We used 11 common inspection items indicating hepatic functions, namely, GOT, GPT, TP, ALB, T-BIL, D-BIL, I-BIL, TTT, ZTT, CHE, and T-CHO. The interval of inspection was approximately one month but varied by patient.

In recent years, interferon (IFN) has been used to cure hepatitis. However, IFN is expensive and has strong side effects. Moreover, the average recovery rate of hepatitis C patients on IFN is only approximately 30%. Therefore, it is crucial to predict the effectiveness of IFN based on the inspection results before IFN is administered. This reduces the physical, mental, and cost burdens on the patient. The data is classified a priori into two classes,

namely, 55 positive examples and 82 negative examples in terms of existence of the hepatitis virus through HCV-RNA inspection before and after IFN administration.
The objective of this experiment is to construct a map that suggests periods of patients group which gives the same trends in attributes under the classes of IFN's effectiveness. The results are validated by comparison with the results obtained by a simple display method given by the ratio of positive and negative examples in each cluster.

6.2 Pre-processing

In order to avoid bias among attributes (inspections), the attribute values were standardized based on the index presented by a doctor. All of the attribute values were standardized in continuous value from 1 to 6. The actual value that is in the range of normal state presented by a doctor was set as 3. Also extremely high and low values were cut off. At the same time, since the inspection intervals are different in each patient, the discretized points are normalized by linear interpolation using one-week intervals. Since the inspection intervals are approximately one month, this interpolation interval is sufficient.

6.3 Visualization Results

In this section, two display methods are compared using the results obtained by the Sequence-based SOM (regular grid: 52x15), i.e., using the same winner neuron trajectories. Figure 10 show the visualization obtained by positive-negative ratio based on Sequence-based SOM, which is obtained by:

$$w_i = \frac{NP_i}{\left(NP_i + NN_i\right)}, \tag{9}$$

where NP_i is the number of positive samples associated with node i and NN_i is the number of negative samples. Figure 11 show the visualization obtained by the single-layer perceptron based on the Sequence-based SOM result. In both maps, the horizontal axis indicates the number days before IFN administration, where the right side represents the administration day. This map indicates that if the winner neuron trajectory passes through high-weight nodes, the period may be related to IFN effectiveness. In addition, an expert can refer to the results for other patients in the same cluster in order to investigate the reason for the findings.
In Fig. 10, since some clusters in 30 weeks before the administration have only a few samples, their node weights tend to zero or one, as indicated by the ratio display method. Therefore, there is a bias to clusters with a small number of samples. This may be mitigated by adding confidence to nodes according to the number of samples. The white nodes connected by white lines indicate an example of positive patient trajectory. In contrast, there is no such bias by the connectivity weights display method. The node values appear to be well balanced as they spread across the entire period and show a larger positive region than the map obtained by the positive-negative ratio.

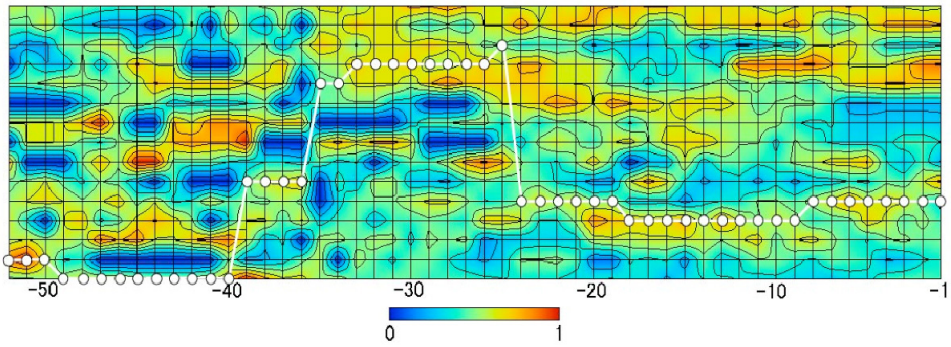


Fig. 10 Visualization by positive-negative ratio based on the Sequence-based SOM result.

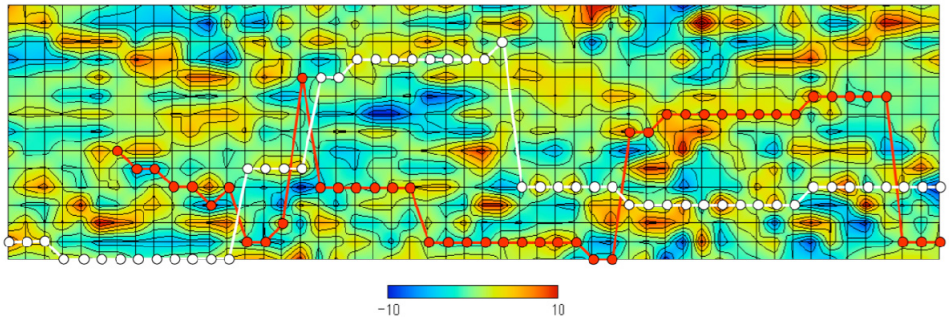


Fig. 11 Visualization by weights obtained by perceptron based on the Sequence-based SOM result.

6.4 Classification Accuracy

We investigated the node values that indicate correct classification, and found that as the weight increased, the correlation with the effectiveness of the IFN also increased. We evaluated the classification performance based on the winner neuron trajectory. In the case of the ratio display method, if the average of the nodes that winner neuron trajectory passes through is greater than 0.5, then the prediction is positive. In the case of the connectivity weights display method, the discriminant function is used as a criterion.

The results of a 10-fold cross validation are shown in Table 3. The average of the maximum and minimum percentages of correct answers obtained by the ratio display method is 47.53% for the test set. The result is equivalent to having a random scheme. Therefore, classification cannot be generalized to predict the effectiveness for an unknown patient. On the other hand, the proposed method provides almost perfect prediction for known data and 64.24% prediction accuracy for unknown patients, albeit with a large variance. The results seem to overfit the training set. Therefore, other methods of estimating appropriate weights, such as maximum entropy, may be considered in the future.

Display method	Positive-negative ratio	Connectivity weights
Training Set	79.78 ± 4.78%	99.60± 0.40%
Test Set	47.53±16.76%	64.29±21.43%

Table 3. Prediction accuracy (10-fold cross validation).

7. Conclusion

In this chapter, we introduced the extension of the SOM to visualize the change of dynamic clusters, movement, range, and merger/separation. The sequence weight function was introduced to the neuron topology in order to introduce temporal order to the topology. Using the proposed Sequence-based SOM, an experiment with news articles revealed transition of topics, level of interest, derivation, and diversification/convergence. In addition, class labels were used to obtain weights in order to display the SbSOM results. We applied this display method to time series of medical data. However, further study is needed about overfitting against the training set.

Acknowledgement

The present study was supported by the Materials Science & Technology Research Center for Industrial Creation (MSTeC) and Kansai Research Foundation for technology promotion (08R010), in Japan.

8. References

Allan, J. (2002). Topic Detection and Tracking: Event-Based Information Organization, Springer, ISBN: 978-0792376644

Andras, P. (2002). Kernel-Kohonen networks, International Journal of Neural Systems, Vol.12, pp.117-135

Barreto, G.A. & Arajo, A.F.R. (2001). Time in Self-Organizing Maps: An Overview of Models, International journal of Computer Research, Special Issue on Neural Networks, Vol.10, No.2. pp.139-179

Belkin, M. & Niyogi, P. (2002). Laplacian eigenmaps and spectral techniques for embedding and clustering, Advances in Neural Processing Systems (NIPS14), pp.585-591

Bishop, C.M. (2006). Pattern Recognition and Machine Learning, Springer-Verlag, ISBN: 978-0387310732

Boulet, R.; Jouve, B.; Rossi, F. & Villa, N. (2008). Batch Kernel SOM and Related Laplacian Methods for Social Network Analysis, Neurocomputing, Vol.71, pp.1257-1273

Fritzke, B. (1994). Growing cell structures: A selforganizing networks for unsupervised and supervised learning, Neural Networks, Vol.7, pp.1441-1460

Fukui, K.; Saito, K.; Kimura, M. & Numao, M. (2006). Visualization Architecture Based on SOM for Two-Class Sequential Data, Lecture Notes in Artificial Intelligence, Vol.4252, pp.929-936, Springer, ISBN: 978-3-540-46537-9

Fukui, K.; Saito, K.; Kimura, M. & Numao, M. (2007). Combining Burst Extraction Method and Sequence-based SOM for Evaluation of Fracture Dynamics in Solid Oxide Fuel

Cell, Proc. of The 19th IEEE International Conference on Tools with Artificial Intelligence (ICTAI), Vol.2, pp.193-196

Fukui, K.; Saito, K.; Kimura, M. & Numao, M. (2008). Sequence-based SOM: Visualizing Transition of Dynamic Clusters, Proc. of IEEE 8th International Conference on Computer & Information Technology (CIT), pp.47-52

Hagenbuchner, M & Tsoi, A.C. (2004). A Supervised Self-Organizing map for Structures, Proceedings IEEE International Joint Conference on Neural Networks (IJCNN), 1923-1928

Ishikawa, Y. & Hasegawa, M. (2007). T-Scroll : Visualizing Trends in a Time-series of Documents for Interactive User Exploration, Lecture Note in Computer Science, Vol.4675, pp.235-246, ISBN: 978-3540748502

Jain, A.K.; Murty, M.N. & Flynn, P.J. (1999). Data Clustering: A Review, ACM Computing Surveys, Vol.31, No.3, pp.264-322

Kimura, M.; Saito, K. & Ueda, N. (2005). Multinominal PCA for extracting major latent topics from document streams, Proceedings of 2005 International Joint Conference on Neural Networks, pp.238-243

Kohonen, T. (2000). Self-Organizing Maps, Springer-Verlag, ISBN: 978-3540679219

Lau, K.W.; Yin H. & Hubbaard, S. (2006). Kernel Self-Organising Maps for Classification, Neurocomputing, Vol.69, pp.2033-2040

Levene, M. & Poulovassilis, A. (2004). Web Dynamics: Adapting To Change In Content, Size, Topology And Use, Springer, ISBN: 978-3540406761

Oian, T.; Srivastava, J.; Peng, Z. & Sheu, P.C.Y. (2009). Simultaneously Finding Fundamental Articles and New Topics Using a Community Tracking Method, PAKDD2009, Lecture Notes in Artificial Intelligence, Vol.5476, pp.796-803

Rossi, F. (2006). Visualization methods for metric studies. Proceedings International Workshop on Webometrics, Informetrics and Scientometrics & Seventh COLLNET Meeting

Roweis, S. & Saul, L. (2000). Nonlinear Dimensionality Reduction by Locally Linear Embedding, Science, pp.2323-2326

Swan, R. & Jensen, D. (2000). TimeMines: Constructing Timelines with Statistical Models of Word Usage, Proc. of the 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp.73-80

Tenenbaum, J.B.; Silva, V. de & Langford, J.C. (2000). A Global Geometric Framework for Nonlinear Dimensionality Reduction, Science, pp.2319-2323

Wang, X. & McCallum, A. (2006). Topics over Time: A Non-Markov Continuous Time Model of Topical Trends, Proc. of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining

8

Visualization with Voronoi Tessellation and Moving Output Units in Self-Organizing Map of the Real-Number System

Yuji Matsumoto, Motohide Umano and Masahiro Inuiguchi
Osaka University
Japan

1. Introduction

The Self-Organizing map (SOM) proposed by T. Kohonen (Kohonen, 1982; 1995), is a type of artificial neural network whose training is unsupervised. It produces a low-dimensional representation of high-dimensional input data, preserving the neighborhood relations as far as possible.
The SOM is difficult to interpret even with such coloring methods as the U-Matrix (Ultsch & Semon, 1990), the P-Matrix (Ultsch, 2003) or cluster connections (Merkl & Rauber, 1997). For interpretable maps, some methods are proposed to place and move the output units on the real-number coordinates plane. For example, Adaptive Coordinates (Merkl & Rauber, 1997) move output units towards the best matching unit but its map does not preserve the topology of the input data, where topology preservation means close vectors in the input space are mapped to nearby locations in output space (Kiviluoto, 1996).
For more interpretable map with preserving the topology, we propose a real-number SOM (RSOM), where we can not only place the output units as point freely on the real-number coordinates plane but also add, remove and moreover move them. Voronoi tessellation visualizes the map of the output units. RSOM is a natural extension of the conventional SOM because Voronoi tessellation for the output units on the square grid generates square regions on the output plane, the same as the conventional SOM. We illustrate several visualization methods such as minimum spanning tree and variable boundary, which help us to understand clusters or relations of input data. We also propose spherical RSOM to illustrate the power of RSOM that can place output units freely on an arbitrary surface.

2. Real-number SOM

The output unit of the conventional SOM is geometrically restricted to be square or hexagon with grid structure in two-dimensional visualization and with geodesic dome in three-dimensional visualization, which result less visual presentation. Fig. 1 shows a map of the conventional SOM with the input data in Table 1 (Kohonen, 1995), which has 16 animal names and 13 attributes.

		Dove	Hen	Duck	Goose	Owl	Hawk	Eagle	Fox	Dog	Wolf	Cat	Tiger	Lion	Horse	Zebra	Cow
	small	1	1	1	1	1	1	0	0	0	0	1	0	0	0	0	0
is	medium	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0
	big	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1
	2 legs	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0
	4 legs	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1
has	hair	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1
	hooves	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1
	mane	0	0	0	0	0	0	0	0	0	1	0	0	1	1	1	0
	feathers	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0
	hunt	0	0	0	0	1	1	1	1	0	1	1	1	1	0	0	0
likes	run	0	0	0	0	0	0	0	0	1	1	0	1	1	1	1	0
to	swim	1	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0
	fly	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0

Table 1. Animal Names and Their Attributes (Kohonen, 1995)

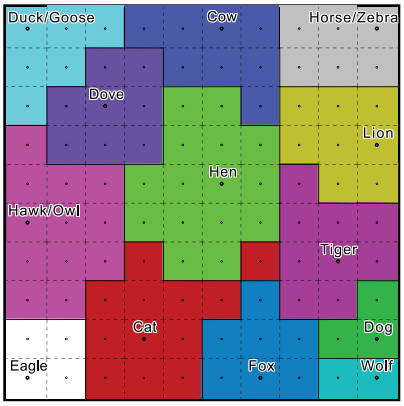


Fig. 1. The conventional SOM trained with input data in Table 1

We propose a real-number SOM (RSOM) whose output units can be freely placed on the real-number coordinates plane. The output unit can be considered as a point rather than a square or a hexagon in the conventional SOM. The visual shapes of these output units on the map of RSOM are determined by the positions of nearby units with Voronoi tessellation. RSOM is a natural extension of the conventional SOM because Voronoi tessellation for the output units on the square grid generates square regions on the output plane, the same as the conventional SOM.

In this section, we describe initialization, training and drawing for RSOM on the two-dimensional plane of $[0,1] \times [0,1]$. Please note that these processes are almost equivalent to the conventional SOM.

2.1 Initial Placement of the Output Units

The output units are placed on the real-number coordinates plane. Random placement is mainly used although many other methods can be applicable.

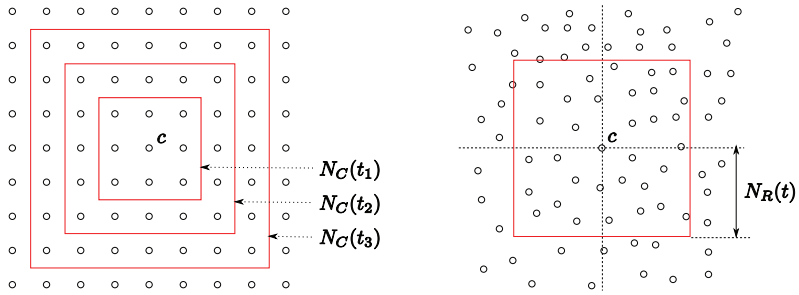
2.2 Training

2.2.1 Selection of the Best Matching Units

We select the best matching unit (BMU) c_i for an input vector $\boldsymbol{x}_i \in \mathbb{R}^n$, which is the smallest of the Euclidean distances of $\boldsymbol{x}_i$ to the reference vector $\boldsymbol{m}_j \in \mathbb{R}^n$ of output unit j:

$$c_i = \arg\min_j ||\boldsymbol{x}_i - \boldsymbol{m}_j||, \tag{1}$$

where $||\cdot||$ is the Euclidean norm and arg min is the value of the argument j for which the value of the given expression $||\boldsymbol{x}_i - \boldsymbol{m}_j||$ attains its minimum value. This step is the same as the conventional SOM.



(a) Conventional SOM (b) RSOM

Fig. 2. Neighborhood Function

2.2.2 Neighborhood Function

The conventional SOM uses discrete coordinates system and defines neighborhood units around the BMU c_i, whose neighborhood function can be defined with Chebyshev distance (i.e., chessboard distance):

$$h_j = \max(N_C(t) - |\boldsymbol{r}_{c_i} - \boldsymbol{r}_j|, 0), \tag{2}$$

where $\boldsymbol{r}_{c_i} \in \mathbb{R}^2$ and $\boldsymbol{r}_j \in \mathbb{R}^2$ are unit locations of the BMU c_i and an unit j on the output layer, respectively, $|\cdot|$ is a chessboard norm ($|\boldsymbol{r}| = \max(x,y)$, where $\boldsymbol{r} = (x,y)$) and $N_C(t) \geq 0$ is some monotonically decreasing function of discrete time t.

Fig. 2-(a) shows the neighborhood function of the conventional SOM, whose rectangle becomes small discretely by time (Kohonen, 1995).

In RSOM, we define a neighborhood function in the same manner as the conventional SOM in the following:

$$h_j = \max(N_R(t) - |\boldsymbol{r}_{c_i} - \boldsymbol{r}_j|, 0), \tag{3}$$

where $N_R(t)$ is a monotonically decreasing function of the time t, e.g., defined as follows:

$$N_R(t) = \max(\delta - \gamma t, 0), \tag{4}$$

where $\delta \geq 0$ and $\gamma \geq 0$ are parameters of neighborhood. Fig. 2-(b) shows the neighborhood function, whose rectangle becomes small smoothly by time.
We can use the neighborhood function of Euclidean norm instead of chessboard norm in Eq. (2): $h_j^E = \max(N_R(t) - ||\boldsymbol{r}_{c_i} - \boldsymbol{r}_j||, 0)$. Gaussian neighborhood function, widely used in the conventional SOM, is also available in RSOM:

$$h_j = \exp\left(-\frac{||\boldsymbol{r}_{c_i} - \boldsymbol{r}_j||^2}{2\sigma^2(t)}\right), \tag{5}$$

where $\sigma(t)$ is the width of the neighborhood at the time t and $||\cdot||$ is the Euclidean norm.

2.2.3 Update

In the same manner as the conventional SOM, reference vectors $\boldsymbol{m}_j$ of output units j in the neighborhood of the BMU c_i are adjusted to the input vector $\boldsymbol{x}_i$ with the following rule:

$$\boldsymbol{m}_j := \boldsymbol{m}_j + \alpha(t)\, h_j \left(\boldsymbol{x}_i - \boldsymbol{m}_j\right), \tag{6}$$

where $\alpha(t)$ is a learning rate at the time t, e.g., defined as follows:

$$\alpha(t) = \alpha_0(1 - t/T), \tag{7}$$

where α_0 is the initial learning rate and T is the number of iteration steps.

2.3 Drawing the Output Layer

The regions of the output units are not explicitly specified since the units are points in RSOM. For visualizing the output layer, we use Voronoi tessellation to divide the output plane into the regions of units, where each point on the output plane belongs to the closest unit. The boundaries of these regions are drawn with dotted line when the dominant classes of two adjacent regions are the same and solid line when these are different. We also put the labels of the dominant class on the BMUs. Fig. 3 shows a map for Table 1 with RSOM, where we place the output units randomly at initial time and use the following parameters: $\alpha_0 = 0.1$, $T = 1000$ and $N_R(t) = 1 - t/T$.
When we place the output units on the square grid, Voronoi tessellation generates square regions on the output plane, like as the conventional SOM, shown in Fig. 1.
Choosing some parameters and functions in RSOM properly, we can have same results as the conventional SOM. Fig. 4 shows other placement patterns of hexagonal grid, spiral, double concentric circle and Sierpinski triangle.

3. Visualization Methods for RSOM

For more interpretable map, we propose two visualization methods such as minimum spanning tree and variable boundary width.

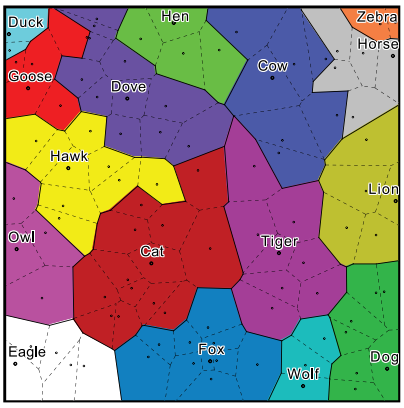


Fig. 3. Voronoi diagram for random placement of RSOM

3.1 Drawing Variable Boundary Width

We can not easily interpret the clusters of "animal family" in Fig. 3. We emphasize the boundaries of two regions whose reference vectors have big difference. When two units j and k are adjacent each other, we draw their boundary with width w_{jk}:

$$w_{jk} = ||\boldsymbol{m}_j - \boldsymbol{m}_k|| \cdot W, \tag{8}$$

where W is multiply factor of boundaries. Fig. 5 shows a Voronoi diagram with variable boundary width for Fig. 3, where boundaries of two similar units are thiner than the ones of different. Fig. 5 is easier to interpret clusters, e.g., *bird*, than Fig. 3.

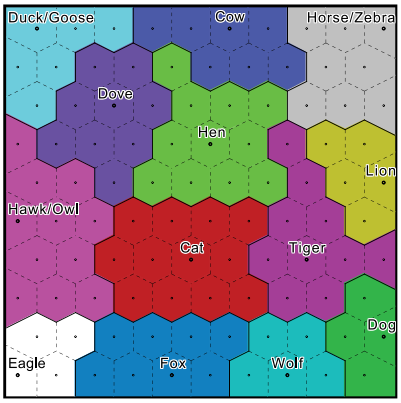
3.2 Drawing Minimum Spanning Tree

Variable boundary width helps us to understand some groups of input. However, this visualization method does not help us to understand relationships between two units very well. For more intuitive understanding, we visualize a tree connecting close units. We have the dual graph for a Voronoi diagram, called the Delauney triangulation whose edges connect adjacent regions in Voronoi diagram. Like Voronoi diagram of RSOM, Delauney triangulation of RSOM shows topological relationship of input vectors in Fig. 6-(a), but this is not easily understandable. Minimum spanning tree in which the weight of each edge is defined in Eq. (8) shows relationships among input vectors like a dendrogram with hierarchical clustering. Fig. 6-(b) shows the minimum spanning tree of Delauney triangulation for the map in Fig. 3, where we can interpret relation of inputs. We can get more understandable maps to use the combination of two visualization methods, variable boundary width and minimum spanning tree in Fig. 7

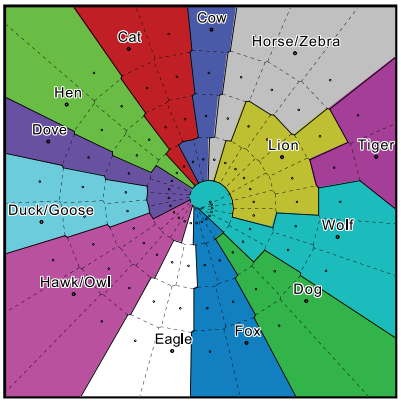
4. Spherical RSOM

Spherical SOM is useful since it removes the "border effect." The conventional spherical SOM uses a geodesic dome restricted the number of output units. We, therefore, propose spherical RSOM, whose output units can be freely placed on a unit sphere.

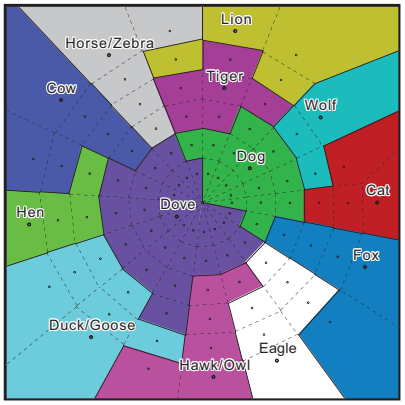
The algorithm for spherical RSOM is the same as that of RSOM except the coordinates systems, the geometrical norm and visualization method of the output layer. For the output units, we



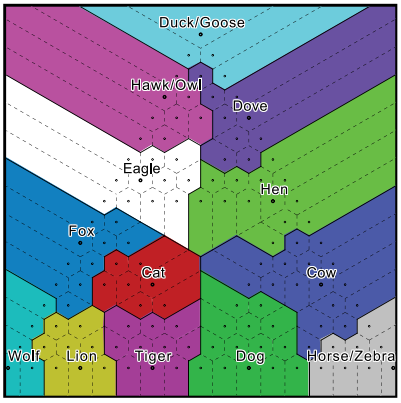
(a) hexagonal grid



(b) Spiral



(c) double concentric circle



(d) Sierpinski triangle

Fig. 4. Voronoi tessellation of hexagonal grid, spiral, double concentric circle and Sierpinski triangle

use polar coordinates $\boldsymbol{s} = (\theta, \varphi)$ ($0 \leq \theta \leq 2\pi$ and $0 \leq \varphi \leq \pi$) and the sphere norm derived from the spherical law of cosines:

$$||\boldsymbol{s}|| = \arccos(\cos\theta, \cos\varphi). \tag{9}$$

Fig. 8 and Fig. 9 show a result of spherical RSOM and its flat map, respectively, where we place output units randomly and use the following parameters: $\alpha_0 = 0.2$, $T = 1000$ and $N_R(t) = 0.5\pi(1 - t/T)$.

Nishio et al. proposed a method for placement on a surface of a sphere with helix (Nishio et al., 2006), in which the number of the units is not restricted, but it does not add, remove and move the units. We can use helix for unit placement of spherical RSOM, and moreover place output units on an arbitrary surface.

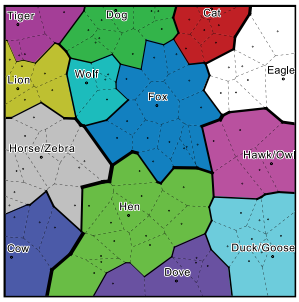


Fig. 5. Voronoi diagram with variable boundary width for Fig. 3

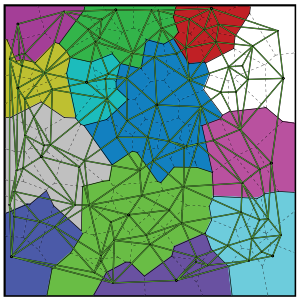
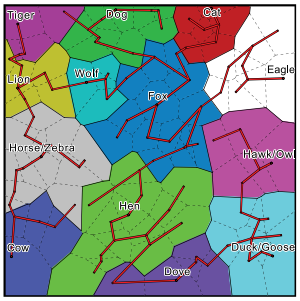


Fig. 6. (a) Delauney triangulation



(b) Minimum spanning tree

5. Related Works

RSOM can not only freely place the output units on an arbitrary surface for visualization, but also add, remove and move the output units for more understandable visualization.
Grawing Grid(Fritzke, 1995a) is a variant of SOM. Grawing Grid initially has 2×2 units and adds a new column (or row) of units when quantization error of a map is high. Grawing Grid considers a some grid network and does not consider move and remove units.
Growing Neural Gas (Fritzke, 1994), (Fritzke, 1995b) uses similar growth technique of Growing Method. Growing Neural Gas can add and remove units but has no feature map. If we have a 2-D map of Growing Neural Gas, we have to use some method, e.g., simulated annealing to create minimum connections(Ogura et al., 2003).

5.1 Visualization Methods

Two visualization methods mentioned above, variable boundary width and drawing minimum spanning tree can be applicable for the conventional SOM.
In the conventional SOM, some visualization methods are developed.
In U-Matrix (Ultsch & Semon, 1990), the relative distances between neighboring reference vectors are drawn in a gray scaled color. The P-Matrix (Ultsch, 2003) visualizes the density relationships in the input space with the Pareto Density Estimation.
In cluster connections (Merkl & Rauber, 1997), connections of the units are represented by a gray scaled color with distances $||\boldsymbol{m}_j - \boldsymbol{m}_k||$.

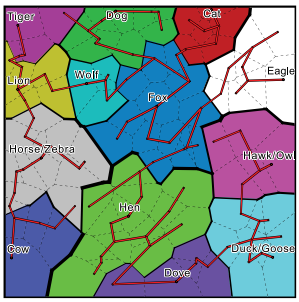


Fig. 7. Variable boundary width overlaying minimum spanning tree for Fig. 5

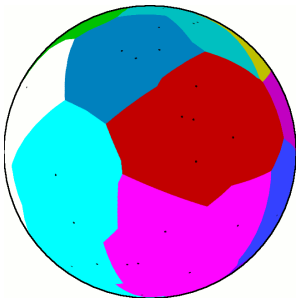


Fig. 8. RSOM on a surface of sphere

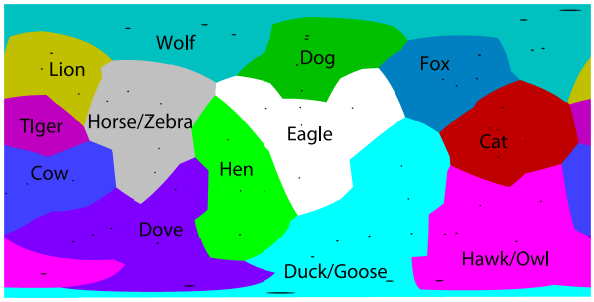


Fig. 9. A flat map of the spherical RSOM in Fig. 8

6. Conclusions

We proposed a real-number SOM (RSOM), which can place output units freely on an arbitrary surface for visualization, with removing and moving the units. We used Voronoi tessellation for visualizing RSOM more expressively. We proposed several visualization methods such as the minimum spanning tree, the variable boundary width, which help us to understand clusters or relations of input data, and the spherical RSOM.

7. References

Fritzke, B. (1994). Fast learning with incremental rbf networks, *Neural Processing Letters* 1(1): 2–5.

Fritzke, B. (1995a). Growing grid—a self-organizing network with constant neighborhood range and adaptation strength, *Neural Processing Letters* 2(5): 9–13.

Fritzke, B. (1995b). *A growing neural gas network learns topologies*, MIT Press, pp. 625–632.

Kiviluoto, K. (1996). Topology preservation in self-organizing maps, *IEEE International Conference on Neural Networks*, pp. 294–299.

Kohonen, T. (1982). Self-organized formation of topologically correct feature maps, *Biological Cybernetics* 43: 59–69.

Kohonen, T. (1995). *Self-Organizing Maps Third Edition*, Springer-Verlag.

Merkl, D. & Rauber, A. (1997). Alternative ways for cluster visualization in self-organizing maps, *Workshop on Self-Organizing Maps, Espoo, Finland*, pp. 106–111.

Nishio, H., Altaf-Ul-Amin, M., Kurokawa, K. & Kanaya, S. (2006). Spherical som and arrangement of neurons using helix on sphere, *Information Processing Society of Japan Digital Courier* 2: 33–137.

Ogura, T., Iwasaki, K. & Sato, C. (2003). Topology representing network enables highly accurate classification of protein images taken by cryo electron-microscope without masking, *Journal of structural biology* 143(3): 185–200.

Ultsch, A. (2003). Pareto density estimation: Probablity density estimation for knowledge discovery, *27th Annual Conference of the German Classification Society*, pp. 91–100.

Ultsch, A. & Semon, H. P. (1990). Kohonen's self-organiring feature maps for exploratory data analysis, *International Neural Network Conference, Dordrecht, Netherlands*, pp. 305–308.

9

Using Self Organizing Maps for 3D surface and volume adaptive mesh generation

Olga Nechaeva
Novosibirsk State University, NSU-Intel Laboratory
Russia

1. Introduction

Adaptive mesh methods are commonly used to improve the accuracy of numerical solution of problems without essential increase in the number of mesh nodes (Lebedev et al., 2002). Within the scope of all adaptive mesh methods, there is an important class of methods in which the mesh is an image under an appropriate mapping of a fixed mesh over a computational domain (Bern & Plassmann, 1999).

Most of widely used conventional methods from the above class, such as equidistribution method (Shokina, 2001), Thompson's method (Thompson et al., 1985), elliptic method (Liseikin, 1999), etc. determine the mapping by solving a complicated system of nonlinear partial differential equations (PDEs). This often leads to significant difficulties. First, the convergence of numerical solution of these PDEs highly depends on an initial mesh, requires fixing boundary mesh nodes beforehand and imposes quite strong limitations on the properties of mesh density function (Khakimzyanov et al., 2001). Second, efficient parallelization of solvers for the PDEs meets overwhelming difficulties. Finally, the PDEs for mesh construction are not universal and need to be proposed for 1D, 2D or 3D spaces specifically. The complexity of numerical solution of these PDEs essentially grows with increasing the dimensionalities (Khakimzyanov et al., 2001). Moreover, there is no methods and techniques in the above mentioned class that can provide a fully automatic adaptive mesh construction in 3D case.

This chapter demonstrates the great ability of the Kohonen's Self Organizing Maps (SOM) (Kohonen, 2001) to perform high quality adaptive mesh construction. Since the SOM model provides a topology preserving mapping of high-dimensional data onto a low-dimensional space with approximation of input data distribution, the proposed mesh construction method uses the same algorithms for different dimensionalities of a physical domain that proves its universality.

In our investigation, the classical SOM model has been studied and modified in order to overcome border effect and provide topology preservation. Based on the ideas in (Nechaeva, 2006), the composition of SOM models of different dimensionalities has been proposed which alternates mesh construction on the border and inside a physical domain. It has been shown that the SOM learning algorithm can be used as a mesh smoothing tool. All the algorithms has been implemented using the GeomBox (Bessmeltsev, 2009) and

GeomRandom (Nechaeva, 2009) packages and tested on a number of physical domains. The quality of resulting meshes is acceptable according to the commonly used quality criteria.
In order to support this alternative approach to mesh generation, a Theorem of Correspondence is proved, that states that goals of traditional PDE approach to construction of adaptive meshes from the considered class are equivalent to the goals of learning for Self Organizing Maps.
The obtained results showed that the neural network approach provides us a highly parallelizable technique (Nechaeva, 2005) for automatic construction of qualitative adaptive meshes and possesses the following properties: (1) due to the self organizing principles the algorithm transforms the mesh automatically, starting with arbitrary initial nodes positions, and does not require to fix the boundary nodes beforehand; (2) stochastic nature of the algorithm enables us to illuminate any limitations on the mesh density function; (3) internal parallelism of the method allows us to parallelize the mesh construction process, taking into account the requirements on the parallel implementation of a problem to be solved on the mesh; (4) the method uses the same algorithms for different dimensionalities of a physical domain that proves the universality of the proposed method.

2. Adaptive mesh construction: problem statement

In order to emphasize that the neural network approach is universal from the point of view of space dimensionality, the problem statement, methods and algorithms are formulated here for arbitrary dimensionalities.
Let G be a physical domain in a Euclidean space R_G^n with physical coordinates $x = (x^1,...,x^n)$. An adaptive mesh $G_N = \{x_1,...,x_N\}$ is to be constructed over G, where $x_i = (x_i^1,...,x_i^n) \in G$, $i = 1,...,N$ are the mesh nodes. Let Q be a computational domain in a Euclidean space R_Q^k, $k \le n$ with coordinates $q = (q^1,...,q^k)$. A mesh $Q_N = \{q_1,...,q_N\}$ is fixed over Q, where $q_i \in Q$, $q_i = (q_i^1,...,q_i^k)$, $i = 1,...,N$. Let a minimal distance among all pairs of nodes in Q_N be equal to d_Q. Usually, the fixed mesh Q_N is rectangular and uniform, then d_Q is just the distance between neighboring nodes. Also, let us denote by $B_\gamma(q)$ a bounded neighborhood of the point q in Q, where γ is a radius of the neighborhood, i.e. $B_\gamma(q) = \{p \in R_Q^k \mid d(p,q) \le \gamma\}$, where $d(\cdot,\cdot)$ is the Euclid distance.
The desired density of an adaptive mesh is given by a mesh density function $w : G \to \mathrm{R}^+$. Density of the mesh G_N is to approximate the function w in a sense of the equidistribution principle (Shokina, 2001). According to this principle, the product of a mesh cell area and the value of w, associated with this cell, should be the same for all mesh cells. As the consequence, the greater the value of w, the smaller the corresponding cell area and, then, the higher the density of the adaptive mesh.
The goal is to find a mapping of Q onto G which transforms the mesh Q_N into the adaptive one G_N with the given mesh density. The method of mapping determination is required to ensure that the boundary nodes of Q_N are automatically transformed into the nodes distributed along the border of G. At this point, the proposed problem statement differs from the traditional one where one needs to have boundary nodes already distributed along

the border (Shokina, 2001). Let Nb be the number of boundary nodes, and Nint be the number of the interior ones, $N_b \cup N_{\text{int}} = \{1,...,N\}$ and $N_b \cap N_{\text{int}} = \varnothing$.

3. Correspondence between adaptive meshes and Self Organizing Maps

The SOM is a neural network model that is able to perform a mapping from input to output space with topology preservation and approximation of input data distribution. When applying the SOM model for adaptive mesh construction, the input space, which contains the physical domain, is R_G^n and the output space is R_Q^k.

The SOM model can be considered as a triplet <*M*, *H*, *Alg*>: map of neurons (*M*), training set (*H*) and learning algorithm (*Alg*). The map of neurons is the set of neurons $M = \{e_1,...,e_N\}$ where each neuron has a location assigned by coordinates of the fixed node q_i in the output space R_Q^k. The number of neurons in the map *M* is equal to the number of mesh nodes *N*. It means that each *i*-th mesh node corresponds to the neuron e_i (Nechaeva, 2004). There is a weight vector associated with each neuron. Weight vector of the neuron e_i is an element of input space and assigned by the coordinates of x_i within the physical domain *G*. These coordinates can be found by the learning algorithm *Alg*. Therefore, the neuron is a pair $e_i = (q_i, x_i)$.

In order to obtain the desired distribution of x_i over *G* in a sense of equidistribution principle, a training set is to be a sample of the probability distribution $p(x)$ being equal to the normalized mesh density function $w(x)$:

$$p(x) = \frac{w(x)}{\int_G w(x)dx}, \tag{1}$$

Let $H = \{y_1,...,y_T\}$ be a training set corresponding to (1), where *T* is the set size, $y_t \in G$, $t = 1,...,T$.

A basic SOM learning algorithm proposed by Kohonen (Kohonen, 2001) (formulated in terms of "mesh nodes"), where $t_{st} = 1$ and $t_{fin} = T$, is the following.

The procedure Alg.

1. Set arbitrary initial locations of mesh nodes $x_i(0)$ for all $i = 1,...,N$.
2. Repeat the following operations at each iteration $t = t_{st},...,t_{fin}$:
   a. Take the next vector y_t from the training set *H*.
   b. Calculate the Euclidean distances $d(\cdot,\cdot)$ between y_t and all nodes $x_i(t)$ and choose the node $x_m(t)$ which is closest to y_t, i.e.

$$m = m(y_t) = \arg\min_{i=1,...,N} d(y_t, x_i(t)). \tag{2}$$

   The node $x_m(t)$ is called a *winner*.
   c. Adjust locations of mesh nodes using the following rule:

$$x_i(t+1) = x_i(t) + \theta_{q_m}(t, q_i)\ (y_t - x_i(t)), \tag{3}$$

for all $i = 1,...,N$, where $\theta_{q_m}(t, q_i) \in [0, 1)$ is a *learning rate*.

At each iteration t, mesh nodes move towards the random point y_t. The magnitude of nodes displacements is controlled by the learning rate $\theta_{q_m}(t, q_i)$.

4. Strategy of learning rate selection

The learning rate $\theta_{q_m}(t, q_i)$ plays a crucial role in the SOM learning algorithm as it directly influences the quality of resulting adaptive meshes and speed of mesh construction.

Unlike usual approach, according to which the algorithm terminates once mesh nodes displacements are small enough, we propose a new strategy where the number of iterations T is to be fixed beforehand proportional to N, and the learning rate is to be scaled in such a way that all nodes are frozen after T iterations. The fact that the number of iterations T should be a function of number N of mesh nodes follows from the point that we need to have enough vectors in the training set for providing each mesh node with a possibility to move several times. For example, we obtained an acceptable quality of adaptive meshes if the number of iterations is 10 times greater than N, i.e. $T = 10N$. It also means that in average each mesh node becomes a winner about 10 times during the iteration process. Therefore, we think that it is incorrect to talk about the number of iterations without mentioning the number of mesh nodes. For example, for our learning rate: $T = 4000$, if $N = 20 * 20$; $T = 90000$, if $N = 30 * 30 * 30$; and so on.

The above technique showed good results in 1D, 2D and 3D cases and is very convenient in use since we can directly control the speed of learning process. The learning rate proposed in this Section is independent of a physical domain and mesh density function.

The learning rate $\theta_{q_m}(t, q_i)$ is a function, which takes its values from the interval [0, 1) and has a view of a product of two functions: learning step and learning neighborhood:

$$\theta_{q_m}(t, q_i) = \delta(t)\ \eta_{q_m}(t, q_i). \tag{4}$$

The learning step $\delta(t)$ is a decreasing function of time and it controls the overall size of nodes displacements at each iteration (Kohonen, 2001), (Fritzke, 1997), (Nechaeva, 2005). Based on experiments, we selected the following function for the learning step: $\delta(t) = t^{-0.2}\chi(t)$, $t = 1, ..., T$, where $\chi(t) = 1 - e^{5(t-T)/T}$. The function $\chi(t)$ is used to make the power member of $\delta(t)$ go down to zero.

Every two neurons q_m and q_i in the map M are connected by a lateral connection with a strength being assigned by the neighborhood function $\eta_{q_m}(t, q_i)$. This function has a shape of Gaussian but is transformed for convenience into the following view:

$$\eta_{q_m}(t, q_i) = s^{\left(\frac{d(q_m, q_i)}{r(t)}\right)^2}, \tag{5}$$

where a constant $s \in (0,1)$ is close to zero and fixed beforehand, $r(t)$ is a learning radius at the iteration t. The shape of the function (5) is shown in Fig. 1.
The strength of lateral connections between the neuron e_m and all neurons $e_i \in M$, located inside the neighborhood $B_{r(t)}(q_m)$ of radius $r(t)$, is less than s. The function for lateral connections (5) satisfies the following properties.

Properties

a. $\forall q_i \in B_{r(t)}(q_m) \Rightarrow \eta_{q_m}(t, q_i) \geq s$.

b. $\eta_{q_m}(t, q_m) = 1$;

c. if $d(q_m, q_i) = r(t)$, then $\eta_{q_m}(t, q_i) = s$;

d. lateral connection is symmetric: $\eta_{q_m}(t, q_i) = \eta_{q_i}(t, q_m)$.

e. $\eta_{q_m}(t, q_i) = \eta(d) = s^{\left(\frac{d}{r(t)}\right)^2}$ is a decreasing function of $d = d(q_m, q_i)$.

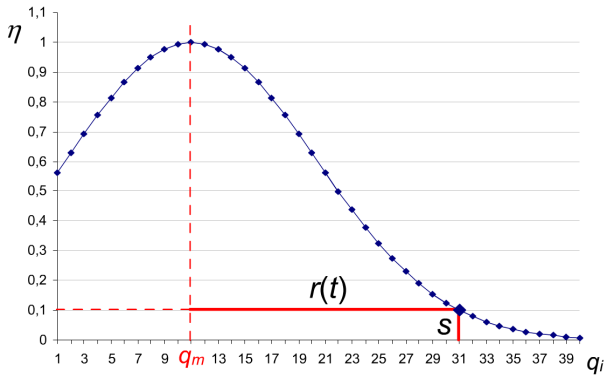


Fig. 1. The shape of the function $\eta_{q_m}(t, q_i)$ for lateral connections between neurons.

As a result, the winner takes the maximum displacement at each iteration. The greater the distance between i-th neuron and the winner in the computational domain, the less the displacement of this neuron within the physical domain. When implementing the SOM algorithm, if s is small enough, neurons for which $d(q_m, q_i) > r(t)$ can be disregarded during adjustment step of the algorithm with preserving the accuracy, since sizes of the displacements are less than $s\delta(t)$.
The function for lateral connections is responsible for the quality of mesh, e.g. mesh smoothness, shapes of mesh cells, etc. The learning radius $r(t)$ is a decreasing function of t with fixed values at first and last iterations: $r(1)$ and $r(T)$, where $r(1) > r(T)$. Based on experiments, we selected the learning radius as $r(t) = r(T) + \chi(t)\left(r(1)0.05^{t/T} - r(T)\right) t^{-0,25}$.
The values $r(1)$ and $r(T)$ has to be selected in such a way that at first iteration all neurons fit into $B_{r(1)}(q_i)$ for any $i = 1,...,N$ and at last iteration the neighborhood $B_{r(T)}(q_m)$ contains only the node q_m and its closest neighbors.

Thus, there are only few free parameters in the proposed learning rate, among them are $r(1)$, $r(T)$ and T. In Fig. 2, diagrams of all functions are shown which take part in the learning rate.

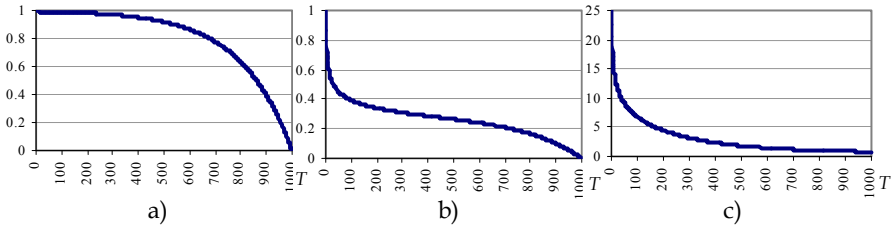


Fig.2. Diagrams of: a) function $\chi(t)$, b) function $\delta(t)$, c) function $\sigma(t)$.

5. Ability of SOM algorithm to order mesh nodes

The whole learning process can be divided into two stages: *ordering stage* and *refining stage* (Kohonen, 2001). During the ordering stage, the learning step and radius are large enough and all mesh nodes takes significant displacements. Therefore, even starting from random initial locations, the mesh nodes become ordered resembling the fixed mesh in the computational domain Q. During the refining stage, the learning step and radius slowly tend to zero and $r(T)$ correspondingly. It leads to a mesh approximating the physical domain in more and more details of a border and density distribution.

In Fig. 3 and Fig. 4, ability of the SOM algorithm to order neuron weights is demonstrated in 2D and 3D cases. During the ordering stage the algorithm tends to reproduce a regular structure of the fixed mesh starting from random initial data.

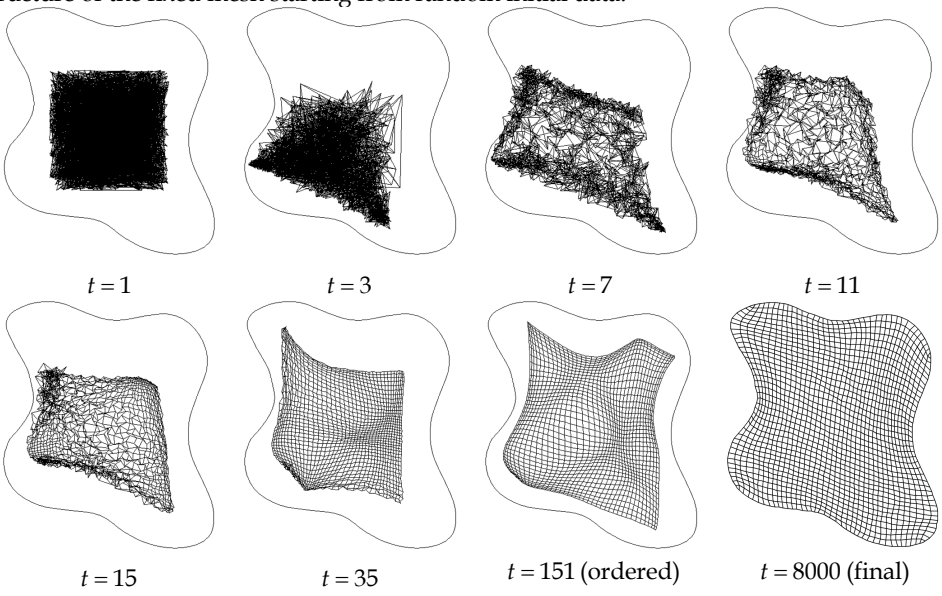


Fig. 3. The ordering stage of SOM learning and final mesh in 2D case.

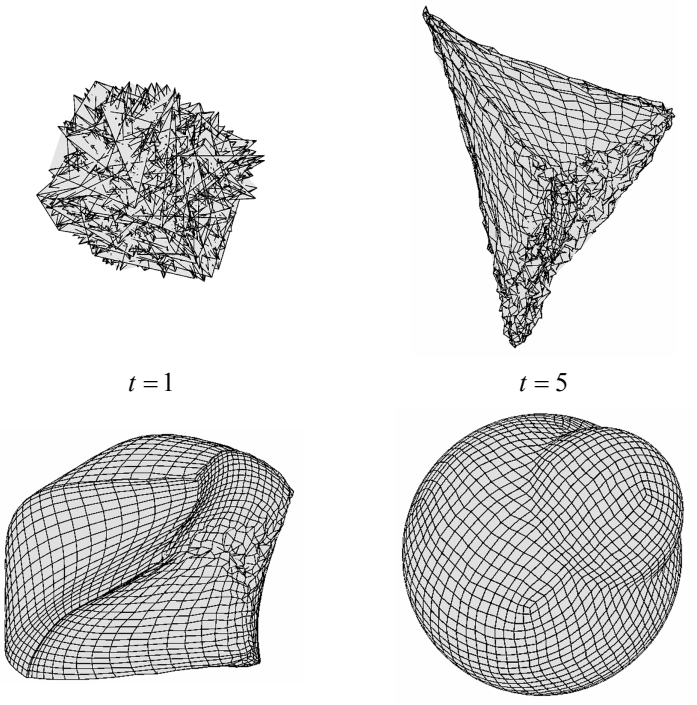


Fig. 4. The ordering stage of SOM learning and final mesh in 3D case.

6. Theorem of correspondence

The SOM learning algorithm aims to find a mapping $m : R_G^n \to R_Q^k$ (more specifically $m : G \to Q_N$) by determining the set of weight vectors $x_1, ..., x_N$. The mapping m has the following form:

$$m(y) = \arg\min_i d(y, x_i) . \tag{6}$$

At best, this mapping has to satisfy the following conditions (goals of SOM learning (Fritzke, 1997)).

(1) *Topology preservation*. If y_i and y_j are near each other in the input space, then neurons $e_{m(y_i)}$ and $e_{m(y_j)}$ are also nearby or $m(y_i) = m(y_j)$.

(2) *Equiwinning Percentage* (EWP). For each *i*-th neuron in the map *M*, there are the same number of vectors y_j in the training set *H* that are closer to the weight vector x_i than to any other weight vector. This also means that each neuron at the end of learning process has the same probability to become a winner for a randomly chosen input vector from *H*.

If these goals are satisfied, it is possible to proof that at the end of the learning process the resulting adaptive mesh reached desired approximation of mesh density function in a sense

of equidistribution principle. This theorem is called the Theorem of Correspondence, since it assigns the correspondence between goals of adaptive mesh construction in a traditional equidistribution sense (formulated in Section 2) and the goals of SOM learning algorithm.
The theorem of correspondence has been formulated and proven in order to show principal possibility to obtain adaptive meshes with given mesh density using SOM. This theorem states that if the EWP goal is reached, then an analogue of equidistibution principle is satisfied for Voronoi cells of the adaptive mesh.
The Voronoi cell V_i is the unbounded set of all point from G closer to x_i than to any other mesh node, i.e. $V_i = \{x \in G \mid d(x, x_i) < d(x, x_k), k = 1,...,N, k \neq i\}$ (Okabe et al., 2000). The whole G then can be represented by closure of the union of disjoint Voronoi cells.

Theorem of correspondence

Let the EWP condition be satisfied for the map of neurons M. Then the product of the square of Voronoi cell V_i and the value of probability density $p(x_i)$ can be estimated by $\frac{1}{N}$ for all i, i.e.:

$$|V_i| \cdot p(x_i) \approx \frac{1}{N}, \quad i = 1,...,N . \tag{7}$$

Proof

Let P_i be the number of elements in the sample $H = \{y_1,...,y_T\}$ which are closer to the node x_i than to any other node, i.e. $P_i = |\{y_j \in H \mid m(y_j) = i\}|$. If there are several closest nodes, the one from such nodes is to be chosen randomly.
According to the definition of integral, the square of V_i can be represented as:

$$|V_i| = \int_G \chi_{V_i}(x)dx , \tag{8}$$

where $\chi_A(x)$ is an indicator of a set A, i.e. $\chi_A(x) = \begin{cases} 1, x \in A \\ 0, x \notin A \end{cases}$. Let us calculate the square of V_i using the Monte Carlo methods (Mikhailov & Voitishek, 2006). After multiplication by the density $p(x)$ of sample H distribution, the integral (8) has the following form:

$$|V_i| = \int_G \chi_{V_i}(x)dx = \int_G \frac{\chi_{V_i}(x)}{p(x)} p(x)dx , \tag{9}$$

and can be considered as an expectation of the stochastic variable having the values $\frac{\chi_{V_i}(x)}{p(x)}$ and being defined over the domain G. Using the sample H, the expectation (9) can be estimated by the finite sum:

$$\int_G \frac{\chi_{V_i}(x)}{p(x)} p(x)dx \approx \frac{1}{T} \sum_{j=1}^{T} \frac{\chi_{V_i}(y_j)}{p(y_j)} . \tag{10}$$

Among all items of the sum (10), there are some which correspond to the elements y_j with the indicator value $\chi_{V_i}(y_j) = 0$. It just means that x_i is not the closest to y_j. The number of nonzero items in (10) is equal to P_i. After simplification, the sum (10) has the following view:

$$\frac{1}{T}\sum_{j=1}^{T}\frac{\chi_{V_i}(y_j)}{p(y_j)} = \frac{1}{T}\sum_{\substack{\alpha=1 \\ y_\alpha \in H\cap V_i}}^{P_i}\frac{1}{p(y_\alpha)}. \tag{11}$$

Further, the values $p(y_\alpha)$ can be approximated by the values $p(x_i)$, since x_i is close to a center of gravity of V_i for the majority of Voronoi cells. It is clear that with $N \to \infty$ the error of such an approximation tends to zero. Finally, we have the following estimation:

$$\frac{1}{T}\sum_{\substack{\alpha=1 \\ y_\alpha \in H\cap V_i}}^{P_i}\frac{1}{p(y_\alpha)} \approx \frac{1}{T}\sum_{\substack{\alpha=1 \\ y_\alpha \in H\cap V_i}}^{P_i}\frac{1}{p(x_i)} = \frac{P_i}{Tp(x_i)}. \tag{12}$$

Since the EWP condition is satisfied, for each *i*-th neuron in the map *M*, there are the same number of vectors y_j in the training set *H* which are closer to the weight vector x_i than to any other weight vector. From this condition it follows that the fraction $\frac{P_i}{T} \to \frac{1}{N}$ with $T \to \infty$. Proceeding to limit in (12), we can get the following:

$$|V_i| \approx \frac{P_i}{Tp(x_i)} \to \frac{1}{Np(x_i)}. \tag{13}$$

After multiplication by $p(x_i)$, we obtain the estimation:

$$|V_i|\, p(x_i) \approx \frac{1}{N}. \tag{14}$$

The estimation (12) is correct for all $i = 1,...,N$.♦

Now, after this theorem is proved, the traditional goals of adaptive mesh construction and ones of the SOM learning can be considered as equivalent. Unfortunately, there is no proof that the goals of SOM learning can always be reached. Moreover, if we apply the basic SOM algorithm, a number of notorious problems often occur leading to the failures of these goals. First, it is impossible to obtain an accurate approximation of border of a physical domain, as it can be seen from the example in Fig. 4 (c) and (d), because boundary nodes never reach the border and they are influenced by the border effect. Second, sometimes, boundary nodes can propagate into the interior of the domain, especially if the probability distribution $p(x)$ is non uniform. That is the result of bad topology preservation as Fig. 6 (a) shows. Finally, the mesh may contain self-crossings (Fig. 6 (c)) that makes it entirely unusable for numerical simulations. In the next Sections, all these problems are considered in details and our solution to them in the form of the composite algorithm is introduced.

7. Border effect evaluation after applying the basic SOM model

The border effect is closely connected with failure of the EWP condition. Thereby, in this section, the EWP condition is evaluated. According to the definition, if the EWP condition is satisfied, then each neuron has the same probability to become a winner. It is convenient to measure it statistically. In other words, the values of function $m(y)$ can be recorded for all vectors of the training set H.

Let a mesh be constructed by a basic SOM algorithm. For evaluation of the EWP condition, a winning statistics has been recorded for the constructed mesh, i.e. how many times each neuron became a winner. It has been found that equal winning percentage directly depends on the final learning radius $r(T)$. If a mesh has been constructed with $r(T)$ being such that the learning neighborhood $B_{r(T)}(q_m)$ contains only the nearest neighbors of q_m, then the winning percentage is almost the same for all neurons. But if $r(T)$ is large, then the adaptive mesh collapses inside the physical domain and, then, boundary nodes become a winner more frequently. In Fig.5, the winning statistics for two different $r(T)$ is shown at the mesh cut. On the other hand, the radius $r(T)$ essentially influences the mesh smoothness in such a way that small radius leads to unsmooth adaptive meshes, and this usually causes the decreasing of the accuracy of numerical simulations on these meshes. The larger the radius, the smoother the adaptive mesh as it is shown in Fig. 5.

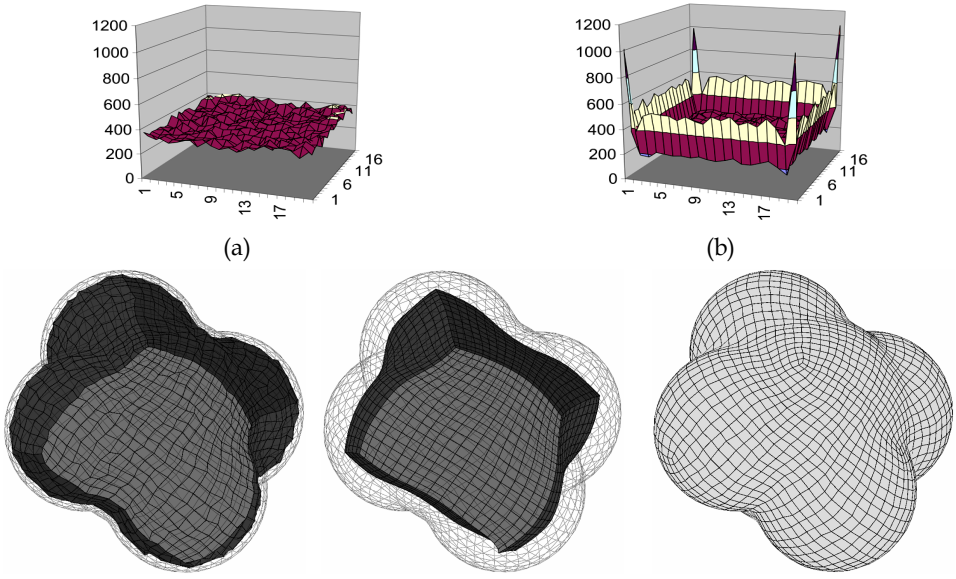


Fig. 5. The winning statistics for the mesh constructed by the basic SOM, mesh size is $20\times20\times20$. (a) $r(T)=1$, (b) $r(T)=10$. The corresponding meshes: (c) $r(T)=1$, (d) $r(T)=10$. The desired mesh (e) is constructed by the composite algorithm proposed in Section 9.

In Section 9, the smoothing algorithm is proposed, which is based on SOM learning with large learning radius and with a technique handling the border effect.
Let us study in details the origins of border effect in the basic SOM model. Being able to measure the border effect, we can handle it. To this end, the iteration number t is assumed to be fixed. Let us consider an interior neuron q_i for which the distance to border of the computational domain is greater than the learning radius r. If the mesh Q_N is rectangular uniform, all neurons q_j from $B_r(q_i)$ as well as all strengths of lateral connections $\eta_{q_j}(q_i)$ are symmetrically located around q_i. Therefore, as it follows from the EWP condition, the neuron q_i has the same probability to be influenced by any other neuron q_j. In the physical domain, it means that the node x_i has the same probability to move symmetrically in all directions being guided by neurons from $B_r(q_i)$. Since s is close to zero, then it is assumed that the mutual influence between neurons q_i and $q_j \notin B_r(q_i)$ is negligibly small.
If the distance from q_i to the border of the computational domain is less than r, then there are not enough neurons in $B_r(q_i)$ for symmetry. In this case, most of the neurons in $B_r(q_i)$ make the neuron q_i move mainly to the center of the physical domain. To balance the asymmetry, the neuron q_i needs to move aside the border of G.
To evaluate the asymmetry, let us consider the following characteristic of the neuron q_i:

$$\alpha_i = \sum_{j=1}^{N} \eta_{q_j}(q_i) . \tag{15}$$

For each node, this characteristic is a sum of lateral connection strengths with all other nodes. If q_i is near the border of Q, then there is not enough terms in the sum (15) corresponding to $B_r(q_i)$. Therefore, α_i is decreasing near the border of Q. It can be clearly seen from the diagram in Fig. 6. All the nodes located at a distance greater than r from the border have the same value of this characteristic.
Obviously, to handle the border effect, it is necessary to balance the asymmetry of lateral connections. It is still an opened question, how the diagrams in Fig. 5 and Fig. 6 are correlated with each other. In the future, this question is going to be answered in order to improve the EWP condition fulfillment.

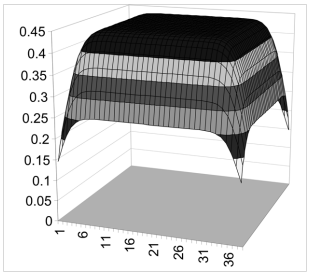


Fig. 6. Characteristic of lateral connections symmetry - values of α_i for the mesh cut, mesh size is $40 \times 40 \times 40$.

It has to be noted that there are some cases when the border effect does not appear. If a map of neurons is closed in such a way that it is not possible to pick out boundary neurons, then this map does not suffer from border effect. Examples of such a maps are: map of neurons forming a ring, map of boundary neurons belonging to a rectangular uniform 2D and 3D grid, a map in the form of torus, and so on. In Fig 5(e), a 3D surface mesh is constructed without border effect.

As a conclusion of this Section, let us point out that even if the EWP condition is fulfilled, the nodes of mesh, obtained by the basic SOM model, still do not reach the border. The explanation of this can be given as follows. Trying to meet the EWP condition, mesh nodes are getting close to the center of gravity of the corresponding Voronoi cell. Since a Voronoi cell is convex, then it is not possible for mesh nodes to appear on the border, at least for convex physical domains. It follows from this that the basic SOM model can be applied *only for interior nodes* when constructing an adaptive mesh. Therefore, in Section 9, the composite algorithm is proposed which is based on the alternative application of the basic SOM models separately to a border and to interior of the domain.

8. Topology preservation after applying the basic SOM model

According to the definition in Section 6, the topology preservation condition is when input vectors that are near to each other in the input space are mapped into nearby or the same neuron locations. As a measure of topology preservation at level γ the quantity $\tau_\gamma \in \mathbb{N}$ can be used, which is defined as follows. Let us consider (together with the winning function $m(y)$) the function of second winner: $m'(y) = \arg \min\limits_{\substack{i=1,\dots,N \\ i \neq m(y)}} d(y, x_i)$. Given $N > 2$ and $\gamma \in \mathbb{R}$, the function $\mu_\gamma(y)$ is defined, which answers the question whether the second winner is in the neighborhood of the first one in *Q* or not:

$$\mu_\gamma(y) = \begin{cases} 0, & q_{m'(y)} \in B_\gamma(q_{m(y)}) \\ 1, & \textit{иначе} \end{cases}$$

Thus, the measure of topology preservation at level γ is a quantity which is equal to the number of training vectors in H, for which the second winner is outside the neighborhood of the first one: $\tau_\gamma = |\{y \in H \mid \mu_\gamma(y) = 1\}|$. It is appropriate to take the value γ in such a way that for each node the neighborhood $B_\gamma(q_i)$ contains the nearest neighbors of q_i. A nonzero value of τ_γ indicates the failure of topology preservation, and non zero values of $\mu_\gamma(y)$ can help to find locations of this failures.

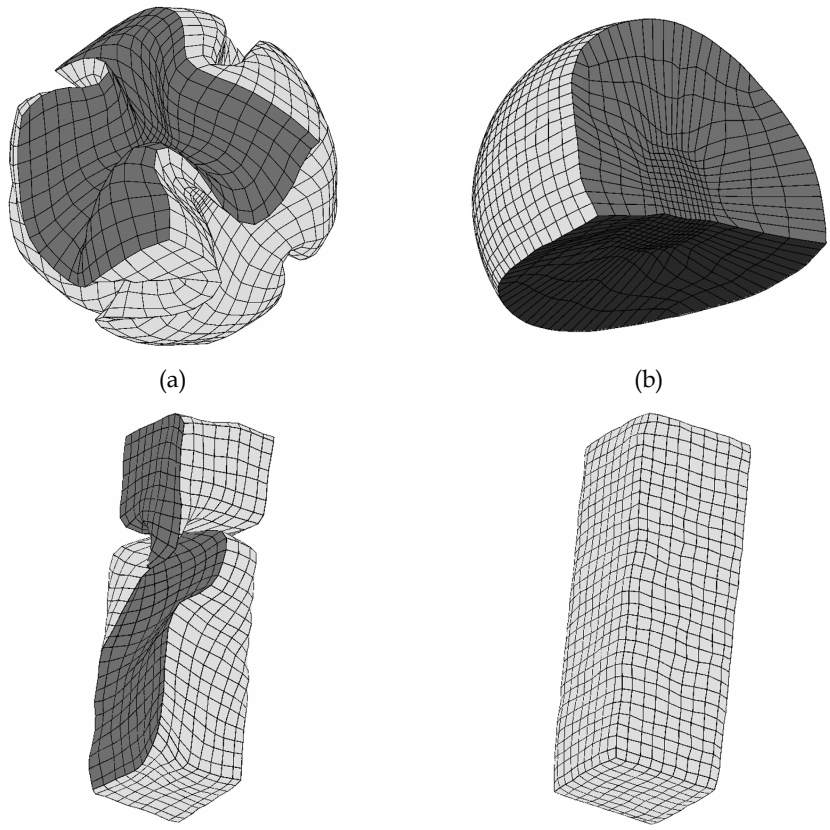
In 3D space, there are a number of typical cases of topology preservation failures when applying the basic SOM model. These cases have equivalents in 2D space too.

1. If the mesh density function is non uniform, then boundary nodes can propagate inside the physical domain being attracted by the high density of input vectors. Usually, such a boundary nodes never leave the attractor and it leads to undesirable bends of the mesh as it is shown in Fig. 7 (a). At this bands, the values of $\mu_\gamma(y)$ are nonzero.

2. If initial locations of mesh nodes are random, there is a probability to obtain a mesh with self-crossings, as it is shown in Fig. 7 (c). But the larger the learning radius $r(1)$, the less the probability of self-crossings. Just because of this at the beginning of the learning process the radius $r(1)$ should cover all the nodes. To further decrease this probability, one can use an initial mesh without self-crossings, for example, rectangular uniform, located somewhere inside the physical domain.

3. When the configuration of physical domain is highly complex, the topology preservation failure can be caused by the inappropriate mesh layout, since its formation is based on self organization. To handle this, the coloring technique can be used, which is described in (Nechaeva, 2007).

All the above cases can be overcome by the composite algorithm.



(a) (b)

(c) (d)

Fig. 7. Topology preservation failures when using the basic SOM model and the desired adaptive meshes constructed by the composite algorithm proposed in Section 9; (a) mesh nodes propagating inside the domain when the mesh density function is non uniform; (c) mesh self crossings.

9. Composite algorithm for adaptive mesh construction

The idea of the composite algorithm (Nechaeva, 2006) is to combine a number of SOM models interacting between each other in a special way and self-organizing over their own set of input data. For example, all neurons in the 3D map M can be divided into two subsets: M_{int} is the set of neurons which correspond to interior nodes and form a 3D volume mesh, and M_b is a set of neurons which correspond to boundary ones and form a 3D surface mesh. In addition, the physical domain G can be divided into a border and interior. Let H_{int} be a training set consisting of vectors only from the interior of G, and H_b is a training set consisting of vectors from the border of G. Taking into account the SOM learning algorithm Alg, we have two SOM models: $\text{SOM}_{int} = < M_{int}, H_{int}, Alg >$ and $\text{SOM}_b = < M_b, H_b, Alg >$. This kind of division seems to be the most convenient for the majority of physical domains which have been studied.

The composite algorithm is based on special alternation of training for each SOM model. The main requirement for the composite algorithm is to provide the consistency between boundary and interior mesh nodes.

Each alternation stage of the composite algorithm consists in training of all SOM models during a given number of iterations, is referred to as a macroiteration, and is denoted by s. For each map M_k, $k = int, b$, there is a private counter of iterations t^k, and the maximum number of iterations T_k is given in such a way that T_k is proportional to $|M_k|$, i.e. to the number of neurons in the map M_k. Let $\varphi_k(s)$ be the number of iterations at the macroiteration s during which the learning procedure is to be applied to the k-th SOM model.

Composite algorithm

(0) Set arbitrary initial weights of all neurons $x_i(0)$, $i = 1,...,N$.

(1) At the first macroiteration ($s = 1$), apply the procedure Alg to the general map M with input vectors taken from H and $t_{st}(1) = 1$, $t_{fin}(1) = T_0$, where T_0 is a given number of iterations.

(2) Repeat the following operations at each macroiteration $s > 1$ until the maximum number of iteration is reached:

   (a) Training of SOM_b. Apply the procedure Alg to the map M_b with input vectors taken from H_b and $t^b_{st}(s) = t^b_{fin}(s-1) + 1$, $t^b_{fin}(s) = t^b_{st}(s) + \varphi_b(s) - 1$.

   (b) Training of SOM_{int}. Apply the procedure Alg to the map M_{int}, but with winner selection from the whole map M. Input vectors are taken from H_{int}. If the winner e_m is from M_b, then replace the input vector $y_{t^{\text{int}}}$ by the weight vector x_m; $t^{\text{int}}_{st}(s) = t^{\text{int}}_{fin}(s-1) + 1$, $t^{\text{int}}_{fin}(s) = t^{\text{int}}_{st}(s) + \varphi_b(s) - 1$.

The step (1) of the composite algorithm is an ordering stage. Application of Alg to all mesh nodes makes the mesh become ordered and take roughly the form of G. The number of iterations T_0 depends on the physical domain configuration. Typically, T_0 is varying from $0.005T$ to $0.01T$. After this step, boundary nodes are located near their appropriate border positions.

The step (2) is a refining stage. Both M_{int} and M_b consistently fit more and more fine details of the interior and border of G. At this stage, at substep (a), boundary nodes have a leading

role. It has been noted that boundary nodes more easily approximate the border than interior nodes approximate the interior of *G*, because in most cases the map M_b is closed (does not have borders). This is true for 3D space as well as 2D space. Therefore, boundary nodes can move within the domain even independently of the interior nodes. At substep (b), interior nodes always follow the boundary ones by means of special winner selection. Since the winner is selected among all the neurons, time to time the winning neuron is a boundary one. In this case, an input vector is replacing by a winner weight vector and all interior nodes move towards the boundary winner. This technique, first, does not let boundary nodes and their interior neighbors propagate inside the physical domain even if there is a subdomain of high density of input vectors; second, keeps a topological connection between interior nodes and their nearest boundary neighbors; and third, excludes mesh self crossings, if there is no self crossings among boundary nodes (that is easy to handle).

We found that the form of functions $\varphi_k(s)$ for defining the number of iterations at each macroiteration is not crucial for the composite algorithm. The resulting mesh is quite the same for different functions $\varphi_k(s)$. For example, these functions can be assigned as $\varphi_k(s) = T_k / S$ for $s > 1$ and $\varphi_k(1) = T_0$, where *S* is the maximum number of macroiterations. But sometimes, it is possible to accelerate the mesh construction by appropriate selection of $\varphi_k(s)$. For example, good results could be obtained if $\varphi_b(s)$ increases and $\varphi_{\text{int}}(s)$ decreases (Fig. 8). The acceleration is achieved because the boundary nodes quickly take correct distribution along the border of *G* and then get frozen giving the interior nodes an advantage until the termination of the composite algorithm. The functions $\varphi_k(s)$ also can be chosen depending on the physical domain configuration.

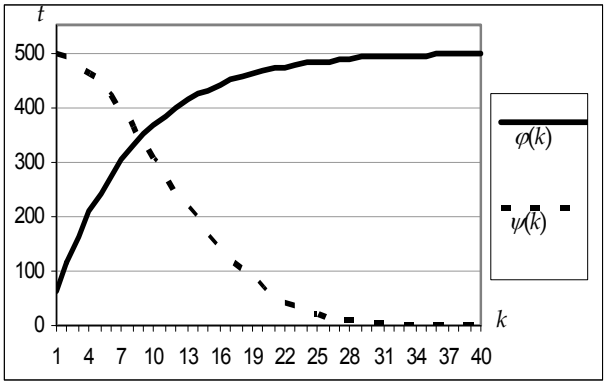


Fig. 8. Diagrams of the functions $\varphi(k)$ and $\psi(k)$.

In Fig. 9, some examples of adaptive meshes constructed by the composite algorithm in 2D and 3D cases are shown.

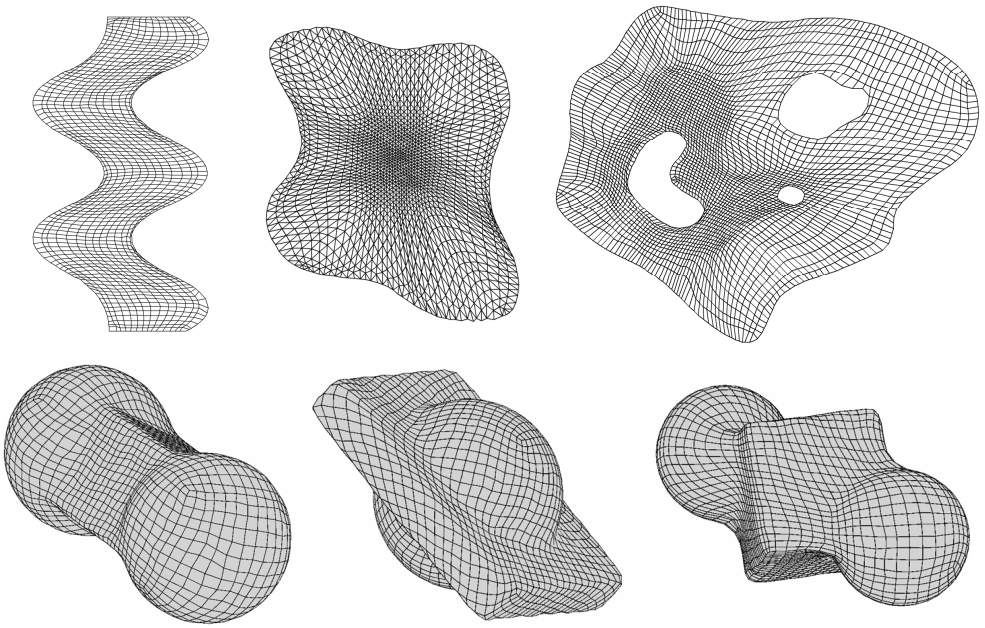


Fig. 9. Examples of meshes constructed by the composite algorithm.

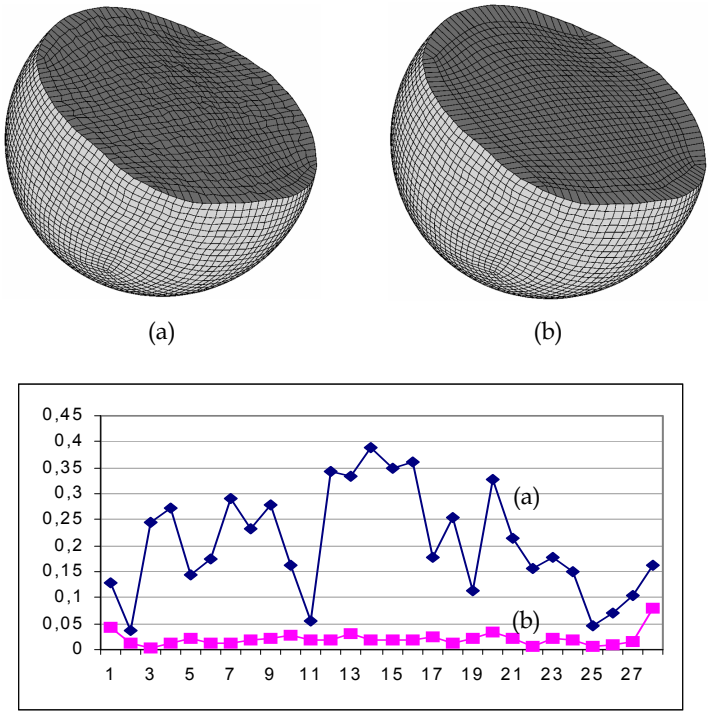
10. Topology preservation after applying the basic SOM model

The composite algorithm overcomes the border effect for small values of learning radius. However, a small radius leads to unsmooth adaptive meshes, and this usually causes the decreasing of the accuracy of numerical simulations on these meshes. The aim of this Section is to propose the technique that allows us to use large learning radius for obtaining the smooth enough adaptive meshes but without the border effect.

Let us consider the case when Q_N is a rectangular uniform mesh. It means that each node of Q_N has four neighbors, and distances between neighboring nodes are equal to d_Q. To measure the smoothness of a quadrilateral adaptive mesh, let us consider a notion of a mesh line which is a set of nodes being the image of a line of the fixed uniform mesh Q_N. Smoothness of a mesh line can be measured by the sine values of angles between two segments, connecting the neighboring nodes in the mesh line, in a sense that the less the quantity of sign inversions and the amplitude of these values, the smoother the line.

Our experiments showed that the mesh smoothness depends on the relation between learning step and radius. Given a fixed learning step, the larger the radius, the smoother the mesh. It can be clearly seen from the example below. In Fig. 10(a), the mesh constructed by the composite algorithm with artificially small final radius $r(T)$ is shown. This mesh is unsmooth even visually. For comparison, in Fig 10(b), the mesh is shown which has been constructed with the final radius 2 times greater. Boundary nodes did not move in this experiment, but they were allowed to become a winner. As it is shown in Fig. 10(c), the smoothness of the last mesh is much better because the sine values are comparatively small

and have less sign inversions. But the resulting mesh is inappropriate for numerical simulations because of bad approximation of the border of the physical domain.



(c)

Fig. 10. Measure of mesh smoothness; (a) the mesh obtained by the composite algorithm with the artificially small final radius $r(T) = 0.5$; (b) the mesh with final radius $r(T) = 6$; (c) diagrams of sine values for the meshes (a) and (b) respectively.

This experiment is a bright demonstration of the border effect in the SOM, which appears when the learning radius is large. The necessary condition for obtaining a smooth enough adaptive mesh is a large learning radius. Therefore, the main problem while smoothing is to handle the border effect.

The general scheme of adaptive mesh construction with employment of the smoothing technique proposed below is as follows. The composite algorithm constructs adaptive mesh with the learning radius being suitable for proper mesh nodes distribution and for fulfillment of the EWP condition. Boundary nodes of this mesh are distributed along the border of G. Starting from this mesh, a SOM-like procedure is applied during the a fixed number of iterations with the constant learning rate, i.e. $r(t) = r$, $\delta(t) = \delta$, $\eta_{q_m}(t, q_i) = \eta_{q_m}(q_i)$, where the learning radius r is comparatively large and the learning step δ is small. This procedure adjusts locations only of the interior mesh nodes and can be regarded as the last stage of the composite algorithm.

After the termination of the composite algorithm, all mesh nodes are distributed over the physical domain according to the given mesh density function. We assume here that the EWP condition is satisfied for this mesh. Therefore, the probability to be a winner is equal to $1/N$.

As it has been shown in Section 7, to eliminate the border effect, it is necessary to balance the asymmetry of lateral connections and to achieve the same value of α_i defined in (15) for all neurons. We propose the technique that allows us to use the boundary nodes as representatives of missing neurons near the border of Q.

Let us imagine that for each boundary neuron, there are K virtual neurons located outside the computational domain. These virtual neurons do not exist in the algorithm but they will help to understand the underlying idea of the proposed technique. The exact locations of virtual neurons are unknown. The only available information is that the distance between k-th virtual neuron and the corresponding boundary neuron q_m is equal to kd_Q, $k = 1,...,K$, where $K = \lceil r / d_Q \rceil$ and $\lceil a \rceil = \arg\min_{n \in \mathbb{Z}} (a < n)$ is the smallest integer no less than a.

To involve virtual neurons into learning process, the following questions are to be resolved: (1) in what conditions a virtual neuron becomes a winner? (2) what are the strengths of lateral connections between neurons q_i, $i = 1,...,N$, and virtual ones? (3) what are the directions and magnitudes of mesh nodes displacements in the physical domain when the winner is a virtual neuron?

Answer to the question (1)

In the case of presence of virtual neurons, winner selection can not be based only on the closeness to the random point, because there are no points outside the physical domain. Therefore, at each iteration, first of all, it is necessary to decide from which kind of neurons the winner is to be selected. Since the EWP condition is satisfied, virtual neurons have the same probability to become a winner as all the other neurons. The probability of virtual neurons to become a winner is equal to $N_b K / (N_b K + N_{\text{int}})$, and hence, an interior neuron can be a winner with the probability $1 - N_b K / (N_b K + N_{\text{int}})$. To select the winner among virtual neurons, an input vector y is selected from the boundary training set H_b, a boundary node which is closest to y is determined, and then the k-th virtual neuron randomly selected (with uniform probability) from the set of virtual neurons, which correspond to the determined boundary node, is assigned to be a winner.

Answer to the question (2)

To define lateral connections strengths between virtual and ordinary neurons, it is necessary to know distances between them in the computational domain. The distance between k-th virtual neuron and the neuron q_i is assumed to be equal to $d(q_m, q_i) + kd_Q$, where q_m is the boundary neuron corresponding to the virtual neuron. This distance is approximate, because the exact location of this virtual neuron in the computational domain is unknown. The lateral connection between k-th virtual neuron related to the boundary neuron q_m and the neuron q_i is taken as follows.

$$\eta_{q_m,k}(q_i) = s^{\left(\frac{d(q_m,q_i)+kd_Q}{r}\right)^2}.$$

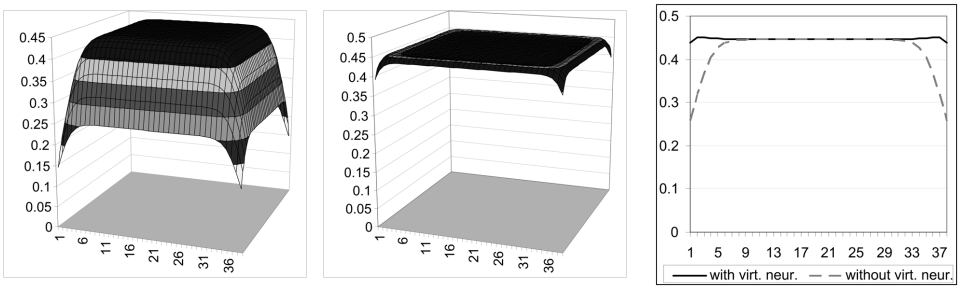
Answer to the question (3)

To specify the directions and magnitudes of the mesh nodes displacements in the physical domain when the winner is a virtual neuron, it is proposed to use the random point y on the border of G, which has been generated for winner selection from the virtual neurons. Let us remind that only interior mesh nodes can move during the smoothing stage. For each interior node x_i, the direction of its displacement is given by the vector $y - x_i(t)$, i.e. the node x_i moves toward the point y located on the border of G. The magnitude of the displacement is equal to $\delta\eta_{q_m,k}(q_i)\cdot v_i(t)\cdot d(y,x_i(t))$, where $v_i(t) = 1 + kd_Q / d(q_m,q_i)$ and q_m is the boundary neuron which corresponds to the virtual winner. This value has been found on the ground of assumption that the ratio between $d(q_m,q_i)$ and $d(y,x_i(t))$ is equal to the ratio between $d(q_m,q_i) + kd_Q$ and $v_i(t)\cdot d(y,x_i(t))$. Since the rule is applied only to interior nodes, then $d(q_m,q_i) \neq 0$.

Taking into account virtual neurons, the characteristic (15) changes and is equal to:

$$\bar{\alpha}_i = \sum_{j=1}^{N} \eta_{q_j}(q_i) + \sum_{k=1}^{K}\sum_{m=1}^{N_b} \eta_{q_m,k}(q_i), \tag{16}$$

where $m = 1,...,N_b$ is an index of a boundary node. In Fig. 11(c), the diagrams of α_i and $\bar{\alpha}_i$ for a mesh cut are shown. It can be seen that $\bar{\alpha}_i$ is almost constant for all neurons in this cut. Therefore, the proposed technique balances the asymmetry of lateral connection near the border.



(a) (b) (c)

Fig. 11. Characteristic of lateral connections symmetry; (a) values of α_i without virtual neurons; (b) values of $\bar{\alpha}_i$ with virtual neurons; (c) the cut of diagrams (a) and (b) where dashed grey line is correspond to (a) and solid black line is correspond to (b).

The algorithm of the smoothing stage is a SOM-like procedure with constant learning parameters. The learning radius r is chosen to be comparatively large, but it is bounded by the curvature of the border of G. The learning step δ is to be small, because fine tuning is needed for smoothing and it does not impair essentially the mesh density approximation.

Besides, when a virtual neuron is a winner, an imaginary random point is outside the physical domain, which can lead to the mesh nodes crossing the border of G. To exclude this

situation, the learning step should satisfy the following condition: $\delta(1+kd_Q/d(q_m,q_i))<1$ for any boundary neuron q_m and interior neuron q_i. From this the following condition can be obtained.

$$\delta < \min_{m,i,k}\left(\frac{d(q_m,q_i)}{d(q_m,q_i)+kd_Q}\right)=\frac{d_Q}{d_Q+Kd_Q}=\frac{1}{K+1}. \tag{17}$$

Smoothing Stage Algorithm

Repeat the following operations during the fixed number of iterations.

1. Generate a random number α from [0,1] with uniform probability distribution.
2. If $\alpha \in \left[0,\ N_{\text{int}}/(N_bK+N_{\text{int}})\right]$, then perform a SOM-like procedure which consists in the following:
   a) Take randomly an input vector y from G using the probability distribution $p(x)$.
   b) Select a winning node $x_m(t)$ among all the neurons. If $x_m(t)$ is a boundary neuron, then replace an input vector with the weights of the winning neuron: $y := x_m(t)$.
   c) Adjust the weights only of the interior neurons according to the rule:

$$x_i(t+1)=x_i(t)+\delta\eta_{q_m}(q_i)(y-x_i(t))$$

3. If $\alpha \in \left(N_{\text{int}}/(N_bK+N_{\text{int}}),\ 1\right]$, then perform the following operations:
   a) Take randomly an input vector y from the border of G with the probability distribution $p(x)|_{\partial G}$.
   b) Select the boundary node $x_m(t)$ which is closest to the point y.
   c) Choose randomly the number k from $\{1,...,K\}$.
   d) Adjust the weights only of the interior neurons according to the rule:

$$x_i(t+1)=x_i(t)+\delta\eta_{q_m,k}(q_i)\left(1+kd_Q/d(q_m,q_i)\right)(x_m(t)-x_i(t)).$$

It has to be again pointed out that virtual neurons do not exist, and thus, there is no need to change the structure of the fixed mesh when counteracting the border effect. Additionally, our efforts have been directed towards the making of the learning rule as simple as possible because of the following reasons: (1) to safe an inherent parallelism of the SOM algorithm which consists in that all neurons are processed according to the same rule independently of each other; (2) to avoid problems when constructing the mesh on a complex multiply-connected domain, i.e. the ones with a single or multiple holes, since the border effect is to be controlled at each of the borders.

10. Quality of resulting adaptive meshes

There are generally accepted quality criteria for quadrilateral 2D and 3D meshes such as the criteria of cell convexity and oblongness, the criterion of mesh lines orthogonality (Prokopov, 1989). In Table 1, possible and admissible values of these criteria

are shown. Also, Table 1 contains the values of criteria for the constructed adaptive mesh, shown in Fig. 9 (b). The values are in the admissible range. Negative values of convexity and orthogonality criteria indicate that there are some non convex cells in the mesh.

Quality criterion	Values Possible/Admissible/Best	Values for Fig. 8(b) Average/Min
Cell convexity	$(-\infty;1]$ / $[0;1]$ / 1	before smoothing 0.825896 / -0.072630 after smoothing 0.927001 / 0.112040
Mesh planes orthogonality (min value of the sin of cell angles)	$[-1;1]$ / $[0;1]$ / 1	before smoothing 0.810640 / -0.034921 after smoothing 0.871268 / 0.158070
Cell oblongness (ratio between max and min edges of a cell)	$(0;1]$ / depending on a problem / 1	before smoothing 0.532211 / 0.000375 after smoothing 0.584676 / 0.098801

Table 1. Mesh quality evaluation.

11. Conclusion

The main result of this investigation is that we proposed an efficient method of adaptive regular mesh construction in 2D and 3D space based on Self Organizing Maps, which does not require solving complicated partial differential equations in order to achieve an acceptable quality of adaptive meshes. The principal possibility of construction adaptive meshes using the SOM models has been proved in the theorem of correspondence.

We believe that the proposed approach to efficient and automatic adaptive mesh construction will contribute to the theory and algorithms of mesh methods. In the future, the neural network approach will be extended to construction of moving structured adaptive meshes based on SOM-like models and unstructured adaptive meshes (Bohn, 1997) based on other self organizing models like Growing Neural Gas and Growing Cell Structures (Fritzke, 1997). The approach seems to be useful especially for building real life geometrical models from point clouds measured by lazer scanners, tomography devices, echo sounding, etc.

12. References

Bern, M. & Plassmann, P. (1999) Mesh Generation, *Handbook of Computational Geometry,* J.-R. Sack and J. Urrutia eds., Chapter 6, Elsevier Science, 1999.

Bessmeltsev, M. (2008) Geometry processing and visualization in scientific applications using GeomBox package, *Electronic publication*, http://aitricks.com/geombox/about.htm, SB RAS, Novosibirsk, Russia, 2009

Bohn, Ch.-A. (1997) Finite Element Mesh Generation using Growing Cell Structures Networks, *Neural Networks in Engineering Systems*, Turku, Finland, 1997.

Fritzke, B. (1997) Some competitive learning methods, *Technical report*, Systems Biophysics, Inst. for Neural Comp., Ruhr-Universität Bochum, April 1997

Khakimzyanov, G.S.; Shokin, Yu.I.; Barakhnin, V.B. & Shokina, N.Yu. *Numerical Modelling of Fluid Flows with Surface Waves,* SB RAS, Novosibirsk, 2001, 394 p.

Kohonen, T. (2001) *Self-organizing Maps,* Springer Series in Information Sciences, V.30, Springer, Berlin, Heidelberg, New York, 2001, 501 p.

Lebedev, A.S.; Liseikin, V.D. & Khakimzyanov, G.S. (2002) Development of methods for generating adaptive grids, *Vychislitelnye tehnologii*, Vol. 7, No. 3, 2002, pp. 29-43

Liseikin, V.D. (1999) *Grid Generation Methods,* Springer-Verlag, Berlin, Heidelberg, New York, 1999, 362 p.

Mikhailov, N.A. & Voitishek, A.V. (2006) *Numerical statistical modeling. Monte Carlo methods*, Publisher: Academia, 2006, 368 p.

Nechaeva, O. (2005) Neural Network Approach for Parallel Construction of Adaptive Meshes, *Lecture Notes in Computer Science, PaCT 2005*, Vol. 3606, Springer, Berlin Heidelberg, 2005, pp. 446-451

Nechaeva, O. (2006) Composite Algorithm for Adaptive Mesh Construction Based on Self-Organizing Maps, *Lecture Notes in Computer Science, ICANN 2006,* Springer Berlin/Heidelberg, Vol. 4131, 2006, p. 445-454

Nechaeva, O. (2007) Composition of Self Organizing Maps for Adaptive Mesh Construction on Complex-shaped Domains, *Proceedings of the 6th International Workshop on Self-Organizing Maps (WSOM 2007)*, Bielefeld University, ISBN: 978-3-00-022473-7, 2007, 6 p.

Nechaeva, O. (2008) GeomRandom package for efficient random nonuniform distribution modeling on a surface and inside complex 3D shapes, *Electronic publication*, http://aitricks.com/geomrandom/about.htm, SB RAS, Novosibirsk, Russia, 2008

Nechaeva, O. I. (2004) Adaptive curvilinear mesh construction on arbitrary two-dimensional convex area with applying of Kohonen's Self Organizing Map, *Neuroinformatics and its applications: The XII National Workshop*, ICM SB RAS, Krasnoyarsk, 2004, pp. 101-102

Okabe, A.; Boots, B.; Sugihara, K. & Sung, N.Ch. (2000) *Spatial Tessellations – Concepts and Applications of Voronoi Diagrams*, 2nd edition, John Wiley, 2000, 671 p.

Prokopov, G. P. (1989) About organization of comparison of algorithms and programs for 2D regular difference mesh construction, *Preprint*, Keldysh Institute for Applied Mathematics, No. 18, Moscow, 1989

Shokina, N. Yu. (2001) Equidistribution Method For Adaptive Grid Generation, *Proceedings of 10th International Meshing Roundtable*, Sandia National Laboratories, 2001, pp.121-133

Thompson, J.F.; Warsi Z.U.A. & Mastin C.W. (1985) *Numerical grid generation, foundations and applications,* North-Holland, Amsterdam, 1985

10

Neural-Network Enhanced Visualization of High-Dimensional Data

Urska Cvek[1], Marjan Trutschl[1] and John Clifford[2]
[1]*Louisiana State University, Shreveport, LA*
[2]*Louisiana State University Health Sciences Center, Shreveport LA*
USA

1. Introduction

Large quantities of multivariate data generated in scientific, engineering, business and other fields triggered exciting developments in information visualization, data mining and knowledge discovery with the objective to identify and describe properties of data sets. Self-organizing map (SOM) is an unsupervised neural network technique capable of analyzing large and complex multivariate data. It attempts to address the problems of high-dimensional data and identify the underlying patterns by reducing the dimensionality achieved through grouping of similar objects and mapping them to a low-dimensional space, usually to a two-dimensional surface also known as a topological map. The results of a SOM could be misinterpreted if taken out of context. For example, the distance between neighbouring weight vectors does not correspond to the physical location of those vectors on the matrix of output nodes as described by Ultsch (Ultsch & Vetter, 1994). The widespread use of the algorithm is attributed to its simplicity. The analytic and graphical Kohonen SOMs and its variations have been successfully applied to the analysis of complex, large-dimensional data sets from diverse sources, including biomedical data, such as by (Durbin & Mitchison, 1990; Tamayo et al., 1999; Van Osdol et al., 1994), just to name a few. The quest for effective and efficient visualization techniques capable of displaying large numbers of high-dimensional records has been formally underway since 1987 when the National Science Foundation sponsored a workshop on Visualization in Scientific Computing as Wong and Bergeron pointed out (Wong & Bergeron, 1997). The problem dates back to the first graphical representation of various types of data sets. Information visualization, as the field is often named, is summarized by the transformation of data – in whatever form – into pictures, with pictures being interpreted by a human being (Spence, 2007). The main advantage of visual displays is the ability to harness the human perceptual system, improving over tabular or other data representation forms.

We utilize the benefits of the SOM and visual displays through a linked technique that integrates the SOM with two- and three-dimensional information visualization techniques (a.k.a., iNNfovis), which serves as a model for constrained self-organization. Properties of iNNfovis environments are harnessed through interactive analysis of large data sets for non-trivial feature extraction. Various iNNfovis configurations provide unique environments for

clustering and exploration of high-dimensional data. Additionally, large-dimensional data mapped to a low-dimensional surface is poised to result in perceptual ambiguities, such as overlap or occlusion, which occurs when the number of records exceeds the number of physical points in a visualization of a certain size. iNNfovis techniques were successfully applied to overcome the effect of occlusion or overplotting by harnessing dimensional information that provides local spatial organization while maintaining a relatively accurate location in a low-dimensional space.
This chapter is organized as follows: Section 2 first provides an overview of a couple of classic visualization techniques used for visualizing multidimensional and multivariate data: scatter plots and RadViz (lossless projection). In Section 3 we discuss the problem of overlap or occlusion in visualizations and how we address it with our technique. Section 4 presents the constrained self-organization technique applied to scatter the plot and RadViz displays. In Section 5 we analyze the transitional cell carcinoma of the bladder using these techniques. We conclude with Section 6.

2. Graphical Representation of Data

Visualization is increasingly used in the data exploration process. In its early years it was mostly, if not only, used to convey the results of statistical computation or data mining algorithms (Chambers et al., 1983; Tukey, 1977; Cleveland & McGill, 1984). Over the last twenty years, its use spread from the exploratory and confirmatory role to the data cleaning process, certain aspects of the data management process and computational steering processes within the data exploration pipeline. There exist numerous data visualization techniques and taxonomies, but the most common data visualization techniques used today are scatter plots, pie charts and line plots. RadViz, star coordinates, polar charts and parallel coordinates are a few of the high-dimensional techniques commonly used by data analysis specialist. Scatter plots, scatter plot matrices, RadViz and the SOM are all projections onto a two-dimensional surface that suffers from occlusion or overlap problem.

2.1 Scatter plot

Scatter plot is a point projection of the data onto a two-dimensional surface or into a three-dimensional space represented on the screen in a classic (x, y) or (x, y, z) format, respectively. Scatter plots can display a large number of points, according to some analysis up to 100,000 points or more, depending on the data pattern (Eick, 2000). Displayed points can have numerous attributes such as color, size, shape, texture, motion and even sound (when interacted with). To interpret the three-dimensional projection of data, it is necessary to resolve ambiguities, although other techniques such as jitter and animation have been used (Chambers, at el., 1983; Chambers & Hastie, 1992). In its most general form this method is related to iconographic and pixel displays. Figure 1 displays the type of Iris flower (Fisher, 1936) as a two-dimensional scatter plot. We project the sepal length dimension as the x axis, and sepal width dimension as the y axis.
Scatter plots reveal a lot of information about the relationship of two variables (distribution of points, outliers, modes, association), but the aspect ratio of the axes, the size and color of points affect their appearance (Ultsch & Vetter, 1994). Scatter plots provide for limited visual scalability, or capability to effectively display large data sets, in terms of the number of elements of the dimension of individual data elements (Eick, 2000). The primary limiting

factor is point overlap: as the number of data instances increases, point over plotting causes increased occlusion of trends and concentration of points, as well as limited access to and hiding of individual instances, causing possible misinterpretation. Approaches that attempt to solve the visual scalability problem include interactivity (focus+context method, panning and zooming, identification and selection, automatic aggregation and brushing), jittering and density estimation. Jittering increases visual scalability by providing access to individual instances, but lacks an insight into the exact relationship among the instances that have been overplotted (Eick, 2000).

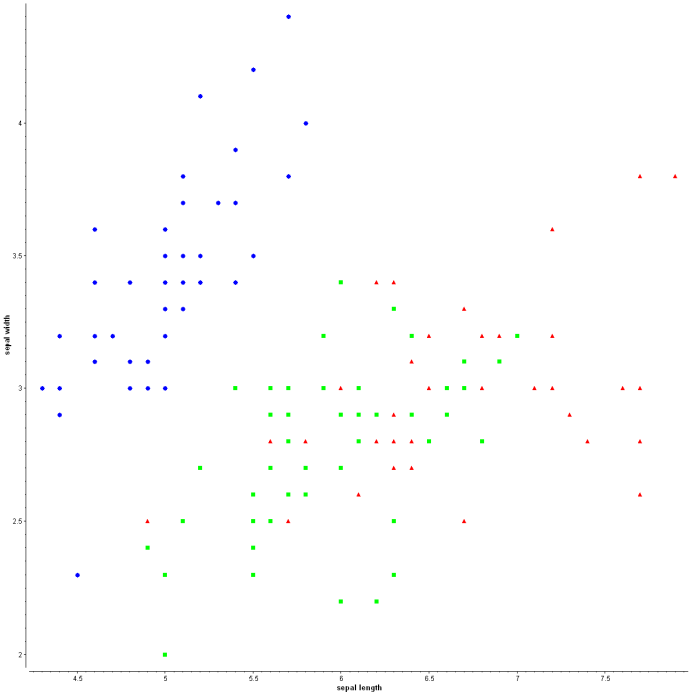


Fig. 1. Two-dimensional scatter plot maps sepal length and sepal width dimensions of the Iris data set

2.2 RadViz

RadViz (Hoffman et al., 1997) is a visualization technique that places dimensional anchors (dimensions) around the perimeter of a circle. Each record is represented as a vector $x_{i1},...,x_{im}$ on these m dimensions and its location in the visualization is determined by the pull of the position vectors (dimensions) $\overline{S}_{1,...,}\overline{S}_m$. Each position vector points from (*0*, *0*) to the corresponding fixed point (dimensional anchor) on the perimeter of the unit circle. Each data point is displayed at the point where the sum of all spring forces equals zero (1).

$$\sum_{j=1,m}\left(S_j - u_i\right)x_{ij} = 0 \tag{1}$$

The position of the data point depends largely on the arrangement of dimensions around the circle; however, data items with similar dimensional values are always placed closer together. The technique has been used in several data domains including biomedical and complemented with dimension ordering techniques, where the dimension order is determined by the structure of the data, or the inherent class separation (Leban et al., 2005; Au et al., 2000; Grinstein et al, 2001; Bertini et al., 2005). We include an example of the technique as applied to the Iris data set, displaying the four dimensions of the data on the circle, and colouring the records by the flower type (Figure 2).

A drawback of the method is overlap of records, that can not only be caused by records that have identical values on the dimensions, but also by records whose sum of all spring forces equals zero (such records are placed in the center of the RadViz unit circle). Overlap of points also occurs when the records are scaled. For example, records (1,1,1,1,1,1) and (10,10,10,10,10,10) would appear at the same location in the center of the circle (they are pulled by all dimensions equally). Records (1,10,1,1,1,1) and (10,100,10,10,10,10) would also appear at the same location. We can encounter instances whose n dimensions have different values, but the sum of their n dimensions is equal to 0. The interpretation of these types of overlaps is more difficult. Dimension ordering and placement of dimensions away from the radial layout minimizes this problem, but does not completely solve it. We developed an approach that utilizes the third dimension to organize the data when overlap occurs to aid occlusion.

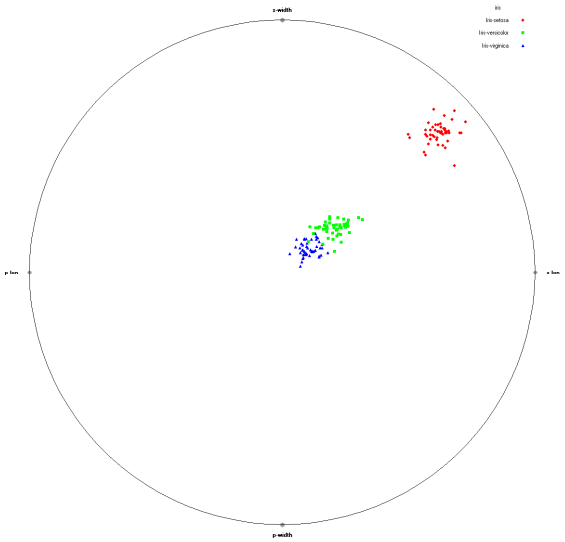


Fig. 2. RadViz visualization of the Iris data set

3. Occlusion or Overplotting in Information Visualization

Large and high-dimensional data sets mapped to low-dimensional visual spaces often result in perceptual ambiguities. One such ambiguity is overlap or occlusion that occurs when the number of records exceeds the number of unique locations in the presentation or when there

exist two or more records that map to the same location. Occlusion in three-dimensional spaces has been extensively studied and approached similarly in two-dimensions (Wong & Bergeron, 1997; Eick, 2000). In these spaces, occlusion is largely due to the placement of records behind the object the user is viewing. The most common approach to resolve it is semi-transparency, where the object is made translucent and additional objects are seen through (Zhai et al., 1996). Overplotted records may also be represented using glyphs where the size of a glyph indicates the number of records that map to the same location (Carr, 1991). In the physical space, where the projection is mostly two-dimensional, transparency of a single object provides only limited occlusion resolution. Another approach is to allow the user to manipulate data, in order to discover additional relationships, such as in the Selective Dynamic Manipulation system (Chuah et al., 1995). Excentric labelling was devised to provide interactive labels for a selection of records by moving the cursor over the records, providing an insight into the values behind these records (Fekete & Plaisant, 1999). Additional techniques were introduced by Manson (Manson, 1999), who utilized retinal properties (size, color and shape), secondary point properties (point border) and animation. The most common approach, used in several commercial packages, is to resolve occlusion using "jittering," displacing overlapping records over a wider area.

Therefore, overlap or occlusion occurs when a data set of n-dimensional records is mapped to an m-dimensional visualization space, where $m < n$ or even $m << n$. Overlap Ω occurs when the number of records r in a data set exceeds the number of physical points in a visualization of size v_x by v_y (in case of a rectangular two-dimensional visualization).

$$\Omega : if\ r > v_x \cdot v_y \tag{2}$$

Overlap may also occur when there are at least two records r_i and r_j that are not unique with respect to their dimensional values x and y or their non-linearly projected x and y positions.

$$\Omega : if\ \exists \left(r_{i_x} = r_{j_x} \wedge r_{i_y} = r_{j_y} \right) \tag{3}$$

Both identities may be modified to reflect the properties of a 3-dimensional overlap.

$$\Omega : if\ r > v_x \cdot v_y \cdot v_z \text{ and } \Omega : if\ \exists \left(r_{i_x} = r_{j_x} \wedge r_{i_y} = r_{j_y} \wedge r_{i_z} = r_{j_z} \right) \tag{4}$$

Overlap can also occur in scatter plots when the two presented dimensions have identical values. Moreover, visualizations based on non-linear projections from an n-dimensional space to an m-dimensional display (such as RadViz and Star Coordinates) most often result in multiple overlapping records regardless of the fact that their dimensional values are unique. All visualizations exhibit similar behavior when analyzing large, high-dimensional data sets. In general, they fail to handle overlapping records and crowding, when mapping numerous instances to a limited display space.

If we assume that we have a two-dimensional visualization, a record maps to position (x, y) on the display. The most common technique to solve the overlap problem is "jittering," which offsets the record from its mapped location (x, y) on the two-dimensional output surface to location (x', y') with $x' = x \pm \Delta x$ and $y' = y \pm \Delta y$, Δx and Δy representing randomly generated offset distances. If more than two points map to the same (x, y) location, the jitter

algorithm randomly generates Δx and Δy for each of the overlapping records, keeping the Δx and Δy within a predefined range. Chambers (Chambers et al., 1983) described jitter in scatter plots that adds random noise to one or both of the variables using two sets of equally spaced values from -1 to 1 and utilizing fractions of the variable range to calculate the offset. Cleveland and McGill (Cleveland & McGill, 1984) and later Cleveland (Cleveland, 1993) described jittering as adding small random variables, in addition to moving points, transformations, open circles and sunflowers as approaches to address overlap.

A more advanced jittering algorithm may also keep track of each jittered point, minimizing possible new overlaps, since it is statistically possible for a random number generator to produce identical offsets Δx and Δy for two or more overlapping points. The amount of jitter can follow a distribution, such as a normal distribution. However, if the number of records is greater than the number of unique locations within the predefined jittering surface of size $\Delta x_{range} \cdot \Delta y_{range}$, the jittering process unavoidably results in new overlaps. Such handling of overlapping records is considered efficient, if the goal is strictly to show the number of records mapping to a particular area, but fails to provide an insight into the exact relationships among the instances that have been overplotted. The main weakness of such jitter technique is spatial displacement that is not driven by the data, causing difficulties in interpretation.

To emphasize the overlap effect, we present overplotting examples of a modified Fisher Iris flower data set that contains four dimensions and the flower type (Fisher, 1936). In this modification, we round the fractional part of sepal length and sepal width dimensional values to the nearest value, and leave petal length and petal width dimensions unchanged. The data set is interesting because in the original data set we cannot find a clear boundary between the three types of flowers when using two pairs of dimensions at a time. Petal width and petal length are very closely related dimensions with only limited correlation existing between those two dimensions and sepal length. In all of our images, we colour Iris *setosa* records red, Iris *versicolor* green and Iris *virginica* blue. Figure 3 is a parallel coordinate display (Inselberg & Dimsdale, 1990) showing the modified set. The result is a reduced number of unique values, leading to multiple overlapping points.

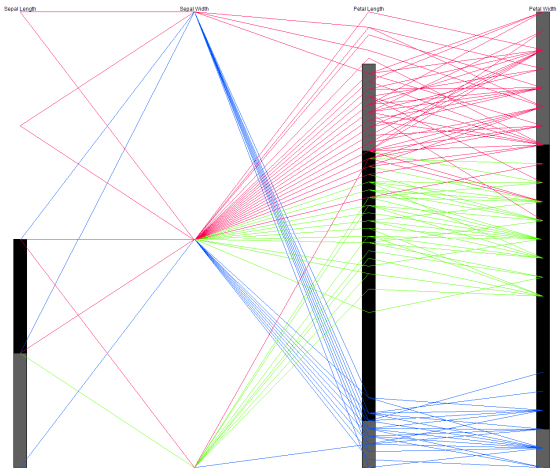


Fig. 3. Parallel coordinates of the modified Iris flower data set

The scatter plot display in Figure 4 uses sepal length and sepal with as the x and y axes, respectively. The visualization contains only eleven points, representing 150 instances, with the remaining 139 instances overlapping at these eleven locations. The overlap is present because the sepal length dimension contains five unique values while the sepal width dimension contains three unique values. This visualization provides little insight into the number of records at each of the locations, or the relationships between individual records. Additionally, the color (type of flower) of each of the points reflects only the last record that was plotted at a location. These records need to be spatially reorganized or jittered in order to address these problems. When the records are randomly jittered, the result displays all or most of the 150 records (Figure 5).

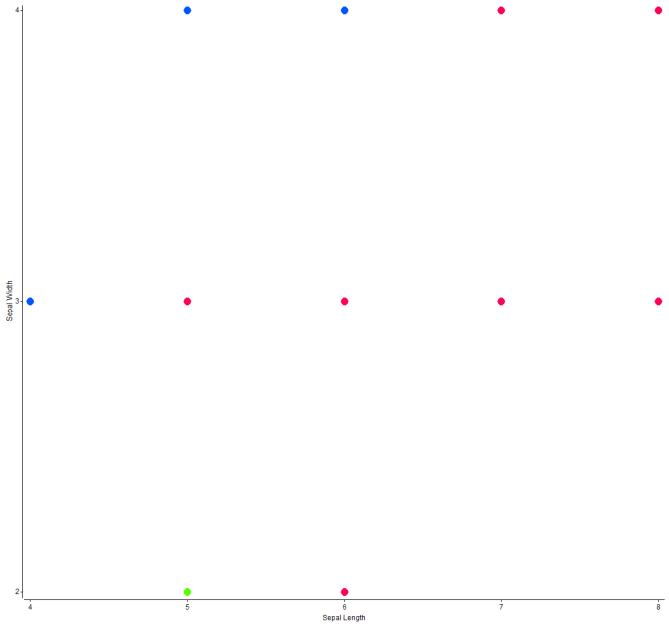


Fig. 4. Scatter plot of the modified Fisher Iris flower; sepal length and sepal width mapped to x and y coordinates. Occlusion results in only one type of flower being displayed at each location.

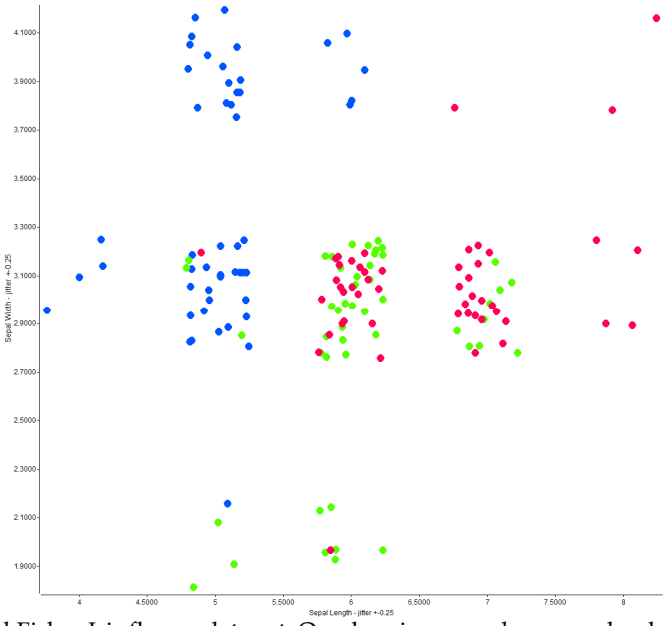


Fig. 5. Modified Fisher Iris flower data set. Overlapping records are randomly jittered to alleviate occlusion. Records jittered between -0.25 and 0.25.

4. Integration of SOM with Classic Data Visualizations: iNNfovis

We utilize the benefits of data visualizations and combine them with the mapping power of the SOM. This combined technique is not a novel way to determine the (x, y) position for a record but rather a refinement method for spatial organization of overplotted input vectors. In general, our approach can be applied to reduce occlusion given any record placement strategy (method to find the x, y position) on a two-dimensional or three-dimensional display. We could use any dimension pair or mapping onto the x and y dimensions, or any other mapping onto a two-dimensional plane or three-dimensional surface. The algorithm harnesses dimensional information of an input vector to provide local spatial organization while maintaining a relatively accurate x and y location on the surface.

The algorithm maps input vectors with similar properties to the same or neighbouring output nodes of the display. Input vectors located in proximity of each other are likely to correlate more than vectors located farther apart. The correlation factor depends on the weight vectors of neighbouring output nodes.

Displacement around a (x, y) position is driven by the dimensions of the data using the SOM algorithm. This creates a constrained clustering environment, a type of unsupervised clustering that is identified by the chosen projection onto the x and y or x, y, z position (three-dimensional) of the output matrix, primary mapping, secondary mapping and the combined SOM-visualization neighbourhood of qualifying output nodes. These constraints differ from constraints in supervised clustering, where the constraints are identified by the outcome of the training process.

4.1 SOM - Scatter plot

We combine SOM with the scatter plot by binning the *x*- and *y-axes* and applying the SOM to this new output grid. Binning is a summarization technique most applicable to large volumes of data and can reduce the number of records to be plotted. Even a scatter plot, for instance, could be treated as an enormous grid of bins, although it is seldom viewed that way. To reflect the properties of the data set, the actual number of bins needed to uniquely map n two-dimensional or three-dimensional vectors is determined based on the range ρ and the number of decimal places δ of data values shown in Equation 5.

$$Bins_{2D} = \left(\rho_x \cdot 10^{\delta_x}\right)\left(\rho_y \cdot 10^{\delta_y}\right) \tag{5}$$

The display surface is replaced with two grids; a secondary output grid within a primary output grid. The grid sizes (or resolution) are specifiable, or can be data-driven, based on the distribution and number of overplotted points, or on the overall number of displayed records.

In Figure 6 we show how we grid the surface of the scatter plot. At the bottom of Figure 6 are the n dimensions of a sample data set, with dimensions 1 and 4 selected as the x and y dimensions. The top of the Figure is the scatter plot grid. Each primary output node on a primary grid contains a set of secondary output nodes, or a grid within a grid. In this example each primary output node contains a secondary output grid of 25 nodes, shaped as a square of 5x5 secondary nodes. We first perform *primary mapping* onto the primary output grid, followed by the secondary mapping. We map a record onto the primary output node *Wp* as determined by the record's values of dimensions 1 and 4. Self-organization is repeated for every primary output node within the grid. *Secondary mapping* first randomly initializes weight vectors of each secondary output node in the primary output node. The distance between an input vector and each output weight vector in this secondary grid is calculated, and the winning secondary output node W_S is determined based on the smallest Euclidean distance. The record is mapped into that winning node W_S (Figure 7) and the weight vector of the winning node adjusted, in addition to limited functional adjustment of the neighbouring secondary weight vectors, depending on the neighbourhood function. Adjustment of weights and self-organization is not limited to a single primary grid, but is rather driven by the neighbourhood function and the properties of the records. This process repeats for every primary output node, in successive training passes through the input data set. The result is a self-organized scatter plot display that addresses the problem of over plotting.

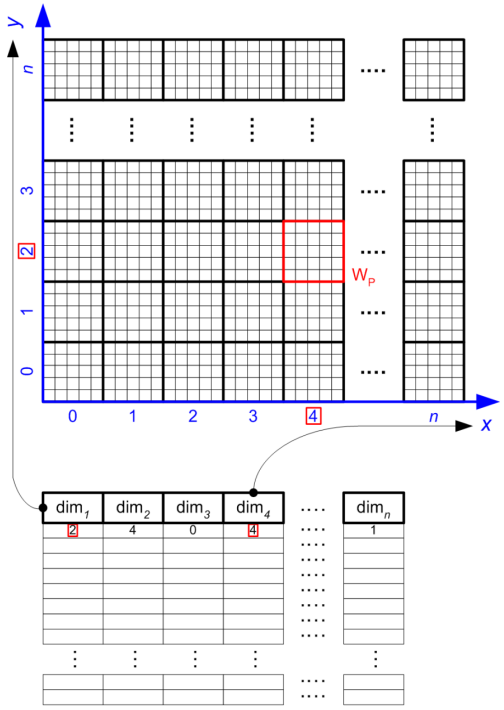


Fig. 6. Secondary output grid within a primary output grid; W_P is one primary output node

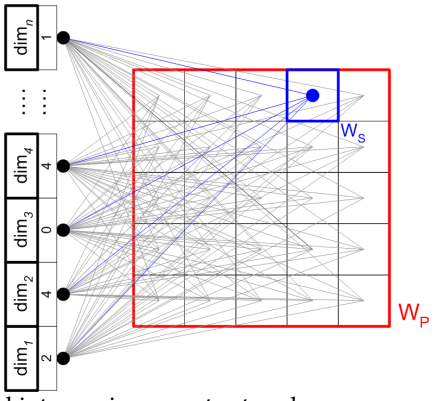


Fig. 7. Mapping of a record into a primary output node

We can similarly extend the two-dimensional scatter plot into the third dimension. We first find the position in the primary grid (R') as we have described, and then select a secondary node in the third dimension into which this record belongs (R''). Figure 8 shows the secondary output nodes available for the mapping. The movement of the record is constrained to the vertical stack of secondary output nodes at the primary grid cell. Each of these secondary output nodes is associated with a weight vector. After the winning

secondary output node is found, a neighbourhood function adjusts the neighbourhood nodes not only in the vertical stack of secondary nodes at the current primary output node, but also for the neighbouring output nodes in the three-dimensional cube, effectively utilizing a three-dimensional Gaussian neighbourhood function. This process is repeated for each of the records in the data set, for a selected number of training epochs.

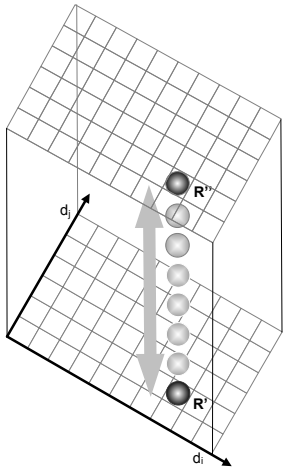


Fig. 8. Secondary mapping into the third dimension (*R''*) from the original position (*R'*) in the primary grid

Figure 9 is a SOM-scatter plot display of the modified Iris data set using a 5x5 primary and 10x10 secondary grid. Each input vector is first placed on the primary grid as determined by the value of sepal length and sepal width dimensions and boundaries of the primary grid of the range of values. Each primary output node contains a 10x10 secondary grid. These 100 secondary output nodes are initialized with random weight vectors, which are adjusted with every input vector mapping. Each input vector is first mapped to the primary output node, and within it into a secondary output node based on the shortest Euclidean distance between an input vector and the weight vectors of the 100 secondary output nodes. The algorithm first maps every input vector until the data set is depleted, and then repeats the training process until the target state is reached. Figure 10 shows a central section of Figure 9 and displaying the lines of separation among the secondary output nodes. These lines are a visual tool for identification of distances among the neighbouring output nodes and extend the U-Matrix (Ultsch, 1994) approach by displaying the records mapping to output nodes combined with inter-nodal distance representations. This feature is a recommended extension for dense primary output grids with large secondary output grid, which take up most of the white space that is otherwise used as an indicator of grid edges. As the lines of separation indicate, it is possible that there is a large difference between two neighbouring secondary output nodes within the same primary output node.

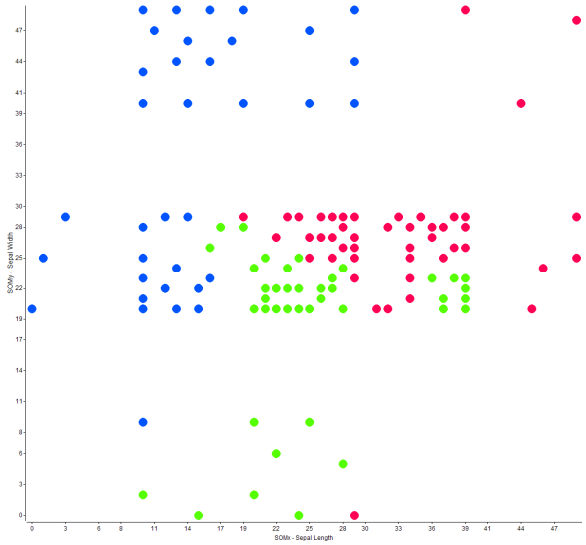


Fig. 9. SOM – Scatter plot visualization of the modified Iris flower data set with 5x5 primary and 10x10 secondary output grid

Fig. 10. Distance between neighbouring output nodes as indicated by the lines of separation. The records shown in colour were selected for further analysis (non-selected records are shown in grey colour).

Records in Figure 10 represent all three flower types. We selected them guided by the lines of separation and selected one *setosa*, three *versicolor* and additional *virginica* records (in color). Parallel coordinates would show that these records differ only on petal length and petal width dimensions. Their values on the sepal length and sepal width dimension are the same, placing them in the same primary output node. SOM-scatter plot separated most of the *virginica* (blue) records from the other four records. Two records within the same primary output node are not necessarily more similar than two records that belong to two different primary output nodes. The *setosa* record in red is more similar to the records in the neighbouring primary output node (grey) than to other records in its primary output node. We also demonstrate that pairs of instances spatially equally distant on the output grid are not necessarily equally similar. The lines of separation show that the weight vectors of secondary output nodes of *virginica* (blue) records are much more similar than the weight vectors of secondary output nodes with *versicolor* and *setosa* records.

4.2 SOM - RadViz

We start the incorporation of the SOM technique with RadViz by binning the surface of the unit circle setup for the RadViz visualization. We determine the position on this primary output grid by applying the RadViz algorithm and selecting the primary output node *R'*, which lies at the position of the RadViz mapping. The user can select the number of grid rows and columns for this RadViz display (see Figure 11), which determines the number of grid nodes. Additionally, we associate each primary output node in this unit-circle grid with a stack of secondary output nodes, to project the input vectors into the third dimension. An identical approach is taken in the three-dimensional scatter plot and this technique significantly increases the intrinsic dimensionality of a RadViz visualization. Our algorithm is capaable of handling the input vectors that exhibit the same profiles that differ in magnitude (i.e., records {1,2,1,2} and {5,6,5,6}). The original RadViz algorithm maps such vectors to the same location thus introducing ambiguity and overlap. Although SOM-RadViz algorithm also places such records at the same (x, y) location, their position on the z-axis differs, where only records with the same or very similar profile are placed in the same secondary output node or near each other.

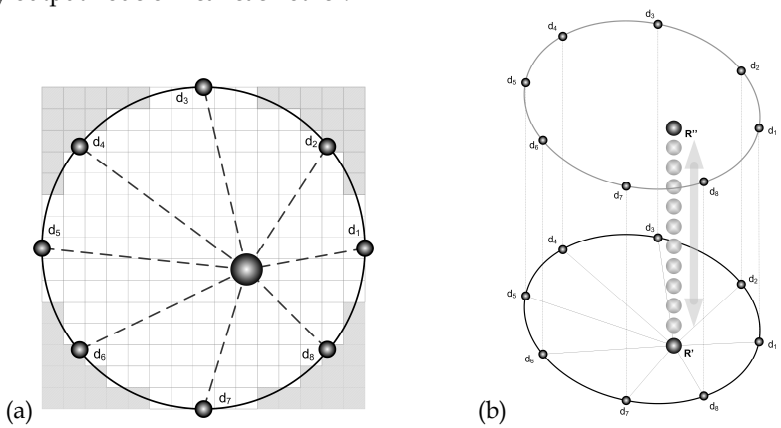


Fig. 11. (a) RadViz mapping with an overlaid 15x15 primary grid; (b) SOM-RadViz mapping into the third dimension

Figure 11 shows this process of first calculating the position of the record in the RadViz display, followed by the "pull" of the record into the third dimension. In this case, we are using twelve output nodes into which the record can map, based on its relationship to other records. The neighbourhood function has effect into the three dimensions, as well.

The modified Iris data set is shown in Figure 12. In this example, we are using a grided surface that replaces the unit circle drawn in the other RadViz displays (simplification). Using the complete four dimensions of this data set, we confirm that *versicolor* and *virginica* flowers aren't separable in two dimensions, but all the records of *setosa* flowers are different in their dimensional values. Please, note that this image is similar to Figure 2, but due to the different zoom of that image, the records appear more separated than in Figure 12. We try to identify other relationships among these records and proceed with the three-dimensional SOM-RadViz approach (right image in Figure 12). In SOM-RadViz, we use the RadViz visualization as the basis (at the floor) separated into 50x50 grid of cells, and then pull the

records into the third dimension of a stack of 50 output nodes at each of these grid locations. We utilize the fairly large grid on RadViz to approximate the true positions of the records as closely as possible. Again, we present it without the unit circle, and rather use the grid as a simplification. Although we previously confirmed that the *setosa* (red) flowers are fairly coherent, we can now detect that their dimensional values identify a structure within them as well. There are approximately four *setosa* records (at the top) that are separated from the rest. There is a cluster of *setosa* records that are not as similar and form a line of records in the center, followed by another set of records at the bottom. Similarly, we now discover that the *versicolor* (green) flowers are very similar and the *virginica* (blue) flowers are separated from them and dissimilar from each other. We reconfirm the relative similarity for the *versicolor* records on all the dimensional values, with some variability. *Virginica* records scatter and are pulled upwards, reflecting the self-organization that the records exhibit based on their dimensional values. This structure is otherwise not seen by any of the other displays – RadViz, scatterplot or SOM and is an excellent example of the power of the combination of the SOM with two-dimensional visualizations.

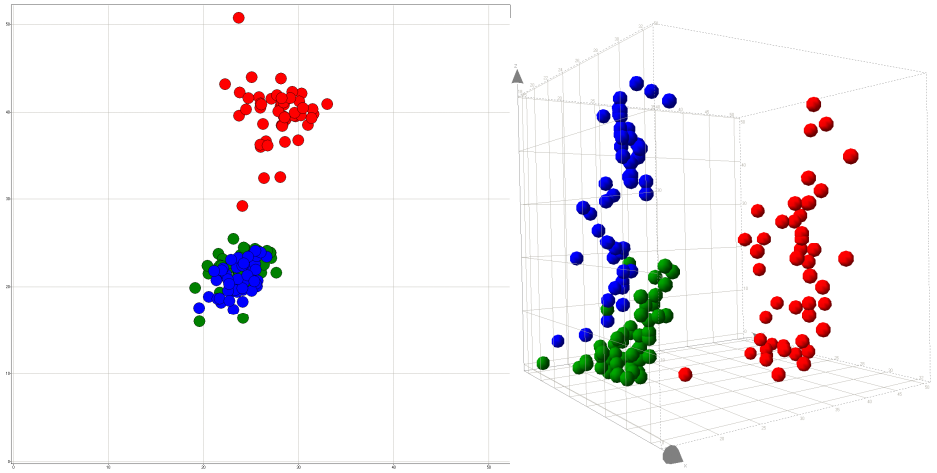


Fig. 12. SOM-RadViz display of the modified Iris data set. The left image is the two-dimensional RadViz projection (as in Fig. 2) and the right image is the three-dimensional SOM-RadViz.

5. Transitional Cell Carcinoma of the Bladder

We are using a data set derived from the analysis of transitional cell carcinoma (TCC) of the bladder generated by the Clifford Lab to demonstrate the application of these techniques (Stone et al.). Transitional cell carcinoma of the bladder ranks 4th in incidence of all cancers in the developed world, yet the mechanisms of its origin and progression remain poorly understood and there are few useful diagnostic or prognostic biomarkers for this disease. In an attempt to generate a mouse model for bladder cancer progression, investigators in the laboratory of Xue-Ru Wu engineered transgenic mice carrying a low copy number of the SV40 large T (SV40T) oncogene, expressed under the control of the bladder urothelium specific murine uroplakin II promoter (Zhang et al., 1999). These mice (called UPII-SV40Tag). develop a condition closely

resembling human carcinoma *in situ* (CIS), a pre-cancerous lesion, starting as early as 6 weeks of age. This condition eventually progresses to invasive TCC from 6 months of age onward. We have combined the UPII-SV40Tag mouse model for bladder cancer progression with Affymetrix DNA microarray technology. With the Mouse GeneChip (Mouse Genome 430 2.0) it is possible to determine a relative expression level of over 39,000 mouse transcripts (45,101 probe sets), representing the majority of the transcribed mouse genome, in a given mRNA sample. Duplicate arrays were performed for UPII-SV40Tag and non-transgenic wild type littermates (WT) at four time points during the course of tumor development, yielding a set of duplicated arrays for two factors: mouse genotype (WT or UPII-SV40Tag) and weeks of age (3, 6, 20, 30) creating eight targets. The WT line at the 6 week time point is the exception; due to the degradation of the RNA we only have one quality array. We followed the recommended analysis techniques (Gentleman & Huber, 2003; Gentleman & Carey, 2005) using R (R Development Core Team, 2008), bioconductor (Gentleman et al., 2004), and used the *limma* (Smyth, 2005) and *affy* packages (Gautier et al., 2004).

The 3, 6, 20 and 30 week time points were chosen to characterize the histologic progression from premalignant urothelium (cells that line the bladder wall), to CIS, to early invasive TCC and finally advanced invasive TCC, respectively, in the UPII-SV40Tag mice. We have identified approximately 1,900 unique genes (as identified by their Affymetrix Probe Set IDs) that are differentially expressed (≥ 3 fold difference at one or more time points) in urothelium between UPII-SVTag mice and their WT age-matched littermates. Preliminary biometric analysis using the Ingenuity Pathways Analysis software package (Ingenuity Systems Inc.) revealed that cell cycle regulatory, DNA replication, and cancer related genes were more strongly expressed in the UPII-SV40Tag urothelium at the highest proportion, even at the 3-week point.

Empirical Bayes method moderated *t*-statistic was used to test each individual contrast equal to zero and compute the moderated F-statistic which combines the *t*-statistic for all the contrasts into an overall test of significance for that gene. The *p*-values were adjusted for multiple testing using the method of Benjamini and Hochberg to control the false discovery rate. Tests were considered to be significant if the adjusted *p* value did not exceed 0.05. We eliminated the control probes from our analysis using the cutoff *p*-value and required at least a one-fold change between the arrays to consider them as differentially expressed.

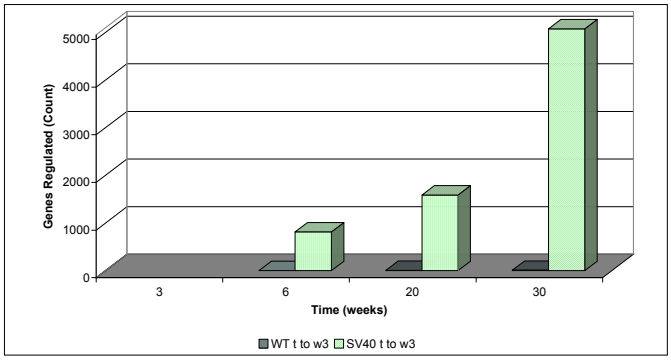


Fig. 13. Gene regulation at time points 6, 20 and 30 for the WT and UPII-SV40Tag lines. Differential expression is observed only in the UPII-SV40Tag line (green).

Figure 13 shows the number of genes that were increased or decreased in expression for each of the lines, when compared to the first time point for the WT. There is virtually no change in expression levels in the WT urothelium, while we observe an exponential increase in the number of differentially expressed genes for the UPII-SV40Tag line from approximately 1,300 to 2,100 and 4,400 at time points 6, 20 and 30, respectively. We next identified the number of genes differentially expressed between the WT and UPII-SV40Tag lines for each time point. Figure 14 shows the number of genes that are differentially expressed at each time point using the F-statistic with an additional requirement of a log fold change of 1.5 or greater. The increase of the number of differentially expressed genes is still apparent, and there are 17 genes that are exponentially changing in expression (either up or down) at every point from 3 weeks to 30 weeks (the intersection).

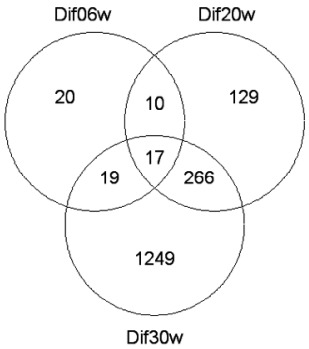


Fig. 14. Two-way analysis confirms the exponential increase in the number of regulated genes with time, comparing UPII-SV40Tag to WT mice at each time point.

A major goal of this project is the identification of biomarkers for early stage bladder TCC. We analyzed the set of 17 genes that change at every one of these time points, hypothesizing that changes at every time point are exacerbated with time progression. At the same time, we are interested in genes that are changing at the early stage of disease progression, namely the time points 6 and 20 (week 20 is classified as the early stage papillary TCC). We then selected a larger set of 585 genes that are differentially expressed at either of the two early stages (Figure 14).

We first analyze the RadViz visualization of this selected set of records. In Figure 15 and following figures we colour the upregulated genes *red* and downregulated genes *green*, following the standard gene expression nomenclature. All the fifteen dimensions of the data set were laid out on the unit circle, starting with the dimensions of the UPII-SV40Tag line followed by the WT line in a counter clockwise fashion. For the case of simplicity, we again use the grided layout. We discern that majority of the genes are upregulated, but the red records prevail in the image, which indicates that a large number of genes are downregulated. These records are pulled into the two directions – to the right side or the left side of the display, depending on the down or up regulation. The positioning of the majority of the records in the centre of the image lead us to believe that most of these records have very similar signal values on the 15 arrays, as the records that pulled away from the centre are more unbalanced and have some values that are out of proportion. We are interested in

the profile of these records and their behaviour over the time course for the two lines (WT and UPII-SV40Tag lines). We utilize the SOM-RadViz visualization as our next step to analyze their behaviour.

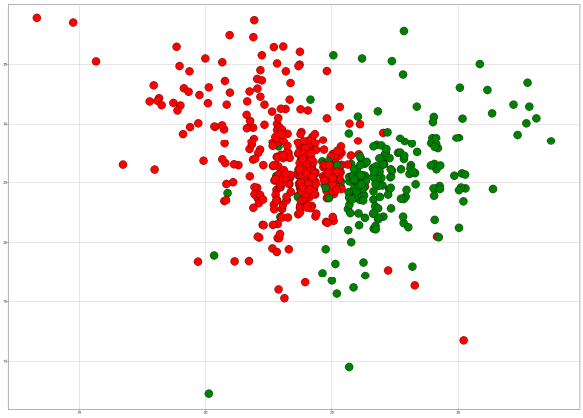


Fig. 15. RadViz visualization of the selected TCC set of 585 Probe Set IDs. Upregulated records are marked red and downregulated green.

We confirm in Figure 16 that a variety of differences exist in the two sets of records. For the SOM-RadViz visualization we use the RadViz approach as described above and overlay the surface of the unit circle using a 50x50 grid of cells. Each of these RadViz cells is provided with a 50-node stack into the third dimension. The neighbourhood function is a three-dimensional Gaussian function, adjusting all the output nodes in the proximity of the mapped location in a three-dimensional space. We used all of the fifteen dimensional values for the RadViz and SOM-RadViz (third dimension) approach. The genes concentrate at the top on the z axis, with much fewer records located at the bottom. We select a set of records from Figure 16 that are located at the three extremes: upper right (7 records) and left (4 records) corners and the bottom (22 records) to showcase the effect of the SOM-RadViz approach and the positioning of the records that is based by their dimensional values.
Figure 17 shows 33 selected records in three colours, depending on their location in Figure 16 (red, blue or purple) and as projected from the camera viewpoint in this figure (side projection). We have to draw a parallel coordinates graph (Figure 18) to show the details of their positioning. The genes in red have low dimensional values on all but the two time points of the UPII-SV40Tag line. The blue genes have higher values on the first two time points of the UPII-SV40Tag line and the purple genes have high dimensional values at all of the time points, although the first two time points of the UPII-SV40Tag line have the least deviation.

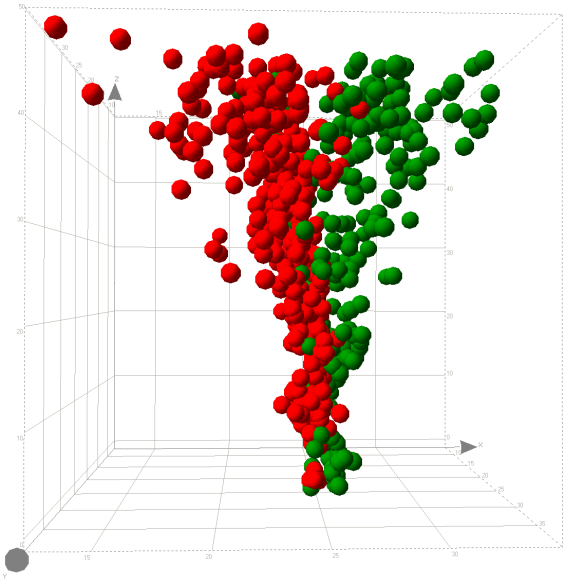


Fig. 16. SOM-RadViz visualization of the selected TCC set of 585 Probe Set IDs

A major goal of this project is the identification of biomarkers for early stage bladder TCC It is hoped that such markers can aid in predicting future developemnt of TCC, particularly in patients that have previously been treated successfully for low grade TCC, but have a high likelihood of recurrence. We have begun testing several of the genes upregulated in UPII-SV40Tag urothelium, including hyaluronan mediated motility receptor (RHAMM), autocrine motility factor receptor (AMFR), proliferating cell nuclear antigen (PCNA) and others as biomarkers for premalignancy, in patient urine samples obtained from a recently completed clinical trial. Our ultimate goal is to establish a panel of markers that can be readily detected in urine that will have high predictive value for TCC, thereby reducing the need for invasive and costly methods such as cystoscopy.

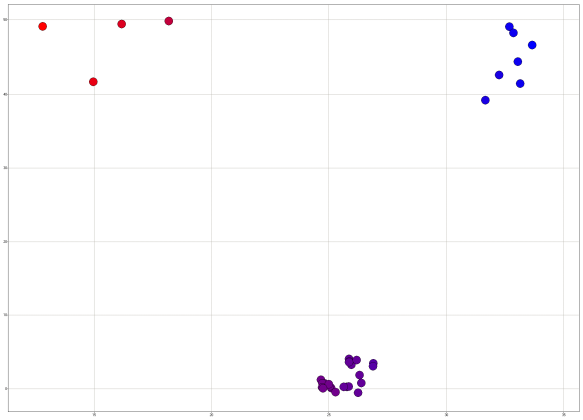


Fig. 17. SOM-RadViz visualization of a subset of 33 records of the TCC set

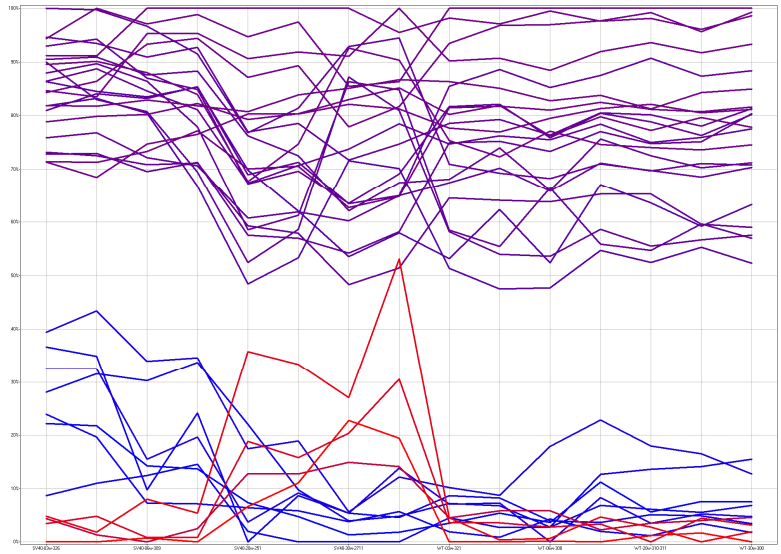


Fig. 18. Parallel coordinates display of the subset of 33 records of the TCC set

6. Conclusion

We provide a description of an approach for intelligent spatial placement of high-dimensional records, based on a modified Kohonen SOM algorithm. SOM-augmented visualizations provide for increased visual scalability and offer higher intrinsic dimension than the classic visualizations. The resulting mapping efficiently alleviates crowding and occlusion, and emphasizes the relationships among the neighbouring multi-dimensional records. Utilizing these techniques, the user can efficiently approach perceptual ambiguities associated with occlusion and gain insight into multi-dimensional data sets using an informative visualization, instead of a series of linked visualizations. These algorithms were tested on high-dimensional biomedical data sets and have provided for meaningful associations in an interactive environment.

7. Acknowledgment

The project described was supported by Grant Numbers P20RR016456 and P20RR018724 from the National Center for Research Resources, and from CA116324 from the National Cancer Institute. The content is solely the responsibility of the authors and does not necessarily represent the official views of the National Center For Research Resources or the National Institutes of Health.

8. References

Au, P.; Carey, M.; Sewraz, S.; Guo, Y. & Ruger, S. (2000). New Paradigms in Information Visualization (poster session), Presented at 23rd International ACM SIGIR Conference, Athens, Greece.

Bertini, E.; Aquila, L. D. & Santucci, G. (2005). SpringView: Cooperation of RadViz and Parallel Coordinates for View Optimization and Clutter Reduction, *Proceedings of Coordinated and Multiple Views in Exploratory Visualization, Third International Conference*, pp. 22-29.

Carr, D. (1991). *Looking at Large Data Sets Using Binned Data Plots: Computing and Graphics in Statistics*. Springer, New York.

Chambers, J.M.; Cleveland, W.S.; Kleiner, B. & Tukey, P.A. (1983). *Graphical Methods for Data Analysis*, Wadsworth.

Chambers, J.M. ; Hastie, T.J. (1992). *Statistical Models is S*, Wadsworth & Brooks/Cole, Pacific Grove, California.

Chuah, M.; Roth, S.; Mattis, J. & Kolojejchick, J. (1995). SDM: Selective Dynamic Manipulation of Visualizations, *Proceedings of ACM Symposium on User Interface Software and Technology*, 61-70.

Cleveland, W.S.; McGill, R. (1984). Graphical Perception : Theory, Experimentation and Application to the Development of Graphical Methods. *Journal of the American Statistical Association*, 79, 387, 531-554.

Cleveland, W.S. (1993). Visualizing Data, Hobart Press, Summit, NJ.

Durbin, R. & Mitchison, G. (1990). A dimension reduction framework for understanding cortical maps. *Nature*, 346, 6259, 644-647.

Eick, S.G. (2000). Visual discovery and Analysis. *IEEE Transactions on Visualization and Computer Graphics*, 6, 1, 44-58.

Fekete, J.-D. & Plaisant, C. (1999). Excentric Labeling: Dynamic Neighborhood Labeling for Data Visualization, *Proceedings of CHI'99*, pp. 512-519, ACM, New York.

Fisher, R.A. (1936). The use of Multiple Measurements on Taxonomic Problems, *Annals of Eugenics*, 7, 179-188.

Gautier, L.; Cope, L.; Bolstad, B.M. & Irizarry, R.A. (2004). affy--analysis of Affymetrix GeneChip data at the probe level. *Bioinformatics*, 12, 3, 307-315.

Gentleman, R. & Huber, W. (2003). Working with Affymetrix data: estrogen, a 2x2 factorial design example. Practical Microarray Course, Heidelberg.

Gentleman, R.; Carey, V.J.; Bates, D.M.; Bolstad, B.; Dettling, M.; Dudoit, S., et al. (2004). Bioconductor: open software development for computational biology and bioinformatics. *Genome Biology*, 5, 10, R80.

Gentleman, R.; Carey, V.; Huber, W.; Irizarry, R. & Dudoit, S. (Eds.) (2005). *Bioinformatics and Computational Biology Solutions Using R and Bioconductor*, Springer.

Grinstein, G.; Jessee, B.; Hoffman, P.; Gee, A. & Oneil, P. (2001). High Dimensional Visualization Support for Data Mining Gene Expression Data, In : *DNA Arrays: Technologies and Experimental Strategies*, CRC Press.

Hoffman, P.E.; Grinstein, G.; Marx, K.; Grosse, I. & Stanley, E. (1997). DNA visual and analytic data mining. *Proceedings of IEEE Visualization 1997*. pp. 437–441, Phoenix, AZ, IEEE Computer Society Press.

Inselberg, A. & Dimsdale, B. (1990). Parallel coordinates: a Tool for Visualizing Multidimensional Geometry, *Proceedings of IEEE Visualization 1990*, pp. 361-378, San Francisco, CA, IEEE Computer Society Press.

Leban, G.; Bratko, I.; Petrovic, U.; Curk, T. & Zupan, B. (2005) VizRank: finding informative data projections in functional genomics by machine learning. *Bioinformatics*, 21, 3, 413-414.

Manson, J. (1999). Occlusion in Two-Dimensional Displays : Visualization of Meta-Data. University of Maryland, College Park, MD.

R Development Core Team. R (2008). *A Language and Environment for Statistical Computing. R Foundation for Statistical Computing*, Vienna Austria.

Smyth, G.K. (2005). Limma: Linear models for microarray data. In: *Bioinformatics and Computational Biology Solutions using R and Bioconductor*. Gentleman, R.; Carey, V.; Huber, W.; Irizarry, R. & Dudoit, S. (Eds.), 397-420, Springer.

Spence, R. (2007). *Information Visualization: Design for Interaction* (2nd Edition), Prentice Hall, Harlow, England.

Stone, R. II.; Sabichi, A.L.; Gill, J., Lee, I.; Loganatharaj, R.; Trutschl, M.; Cvek, U. & Clifford, J.L. Identification of genes involved in early stage bladder cancer progression. *Unpublished*.

Tamayo, P.; Slonim, D.; Mesirov, J.; Zhu, Q.; Kitareewan, S.; Dmitrovsky, E.; Lander, E. S. & Golub, T.R. (1999). Interpreting patterns of gene expression with self-organizing maps: methods and application to hematopoietic differentiation. *PNAS*, 96, 6, 2907-2912.

Tukey, J. W. (1977). *Exploratory Data Analysis*. Addison-Wesley, Reading, MA.

Ultsch, A. (1994). The Integration of Neural Networks with Symbolic Knowledge Processing. In : *New Approaches in Classification and Data Analysis*, Diday et al, (Ed.), 445-454, Springer Verlag.

Ultsch, A. & Vetter, C. (1994). Self-Organizing-Feature_maps versus Statistical Clustering : A Benchmark. *Technical Report*, 9, Department of Mathematics, University of Marburg, Marburg, Germany.

Van Osdol, W. W.; Myers, T. G.; Paull, K. D.; Kohn, K. W. & Weinstein, J. N. (1994). Use of the Kohonen self-organizing map to study the mechanisms of action of chemotherapeutic agents. *J. Natl. Cancer Inst*, 86, 24, 1853-1859.

Wong, P. & Bergeron, R. (1997). Scientific Visualization – Overviews, Methodologies and Techniques : 30 Years of Multidimensional Multivariate Visualization, In : *Scientific Visualization, Overviews, Methodologies, and Techniques*, 3-33, IEEE Computer Society Press, 0-8186-7777-5, Los Alamitos, CA, USA.

Zhai, S.; Buxton, W.; Milgram, P. (1996). The Partial-Occlusion Effect: Utilizing Semitransparency in 3D Human-Computer Interaction, *ACM Transactions on Computer-Human Interaction*, 3, 3, 254-284.

Zhang, Z.T.; Pak, J.; Shapiro, E.; Sun, T.T. & Wu, X.R. (1999). Urothelium-specific expression of an oncogene in transgenic mice induced the formation of carcinoma in situ and invasive transitional cell carcinoma. *Cancer Res.*, 59, 14, 3512-3517.

11

THE SELF-ORGANZING APPROACH FOR SURFACE RECONSTRUCTION FROM UNSTRUCTURED POINT CLOUDS

Vilson L. DalleMole, Renata L. M. E. do Rego and Aluizio F. R. Araújo
Federal Technologic University of Paraná, Federal Institute of Pernambuco, Federal University of Pernambuco
Brazil

1. Introduction

Surface reconstruction from a point cloud is very useful in many different application areas, such as manufacturing (Bernardini *et al.*, 1999), cultural heritage (Bernardini *et al.*, 2002) and medicine (Satava & Jones, 1998). Surface reconstructions methods aim to create digital models to reproduce an object shape given a set of points sampled from its surface using 3D scanning technology.

Surface reconstruction starting from a cloud of points is a complex problem which raises a number of challenging issues: Firstly, the connectivity between the vertices must be constructed so that the reconstructed surface has the same topological features as the target surface. However no structural information is available in the input data. Instead the only items of information available are the 3D coordinates of a set of points sampled from the target surface. Secondly, meshes with different resolutions must be generated to fulfil the needs of different applications, otherwise additional processing is required to simplify (or refine) the mesh constructed. Another issue is that the meshes produced must be two dimensional manifolds. Finally, the triangular faces of the mesh should be approximately equilateral.

A lot of research effort has been expended to develop surface reconstruction methods. Some of these methods are based on geometric techniques (Amenta *et al.*, 2001), (Hoppe *et al.*, 1992). Another well known approach is that of dynamic methods (Miller *et al.*, 1991), (Qin *et al.*, 1998), based on the evaluation of energy or force functions. A more recent approach to the problem of surface reconstruction is that of learning-based methods. Learning algorithms are able to process very large and/or noisy data, such as point clouds obtained from 3D scanners and have been used to reconstruct surfaces. Following this approach, some studies (Brito *et al.*, 2008), (Hoffmann & Varady, 1998), (Yu, 1999), have employed Self-Organizing Maps (SOM) and their variants for surface reconstruction. SOM is suitable for the surface reconstruction problem because it can form topological maps to replicate the distribution of input data. In this chapter the authors present two Self-Organizing Maps based on what that they have proposed for surface reconstruction.

The rest of this chapter is structured as follows: Section 2 presents the necessary background for understanding the surface reconstruction methods presented in Sections 4 and 5. The self-organizing approach for surface reconstruction is presented in Section 3, where we discuss some of the self-organizing models useful for surface reconstructions. In Section 4 and 5 we present, respectively, the GSRM and GSOSM methods for surface reconstruction. Section 6 concludes this chapter.

2. Background

The geometric pipeline presented in (Saleem, 2003) describes the problem of reconstructing computational models of real object surfaces (called target objects). The first step of the pipeline consists of digitalizing the target object to get a set of points sampled from its surface. When no information about the connectivity among the points is available, or rather, the only items of information available for the reconstruction are the 3D coordinates of the sampled points, the output of the first step of the pipeline is called an unstructured point cloud. When additional information about the connectivity among the points is available, the point could is deemed to be structured. This study does not investigate the techniques available for digitalizing real objects, since the focus here is on the task of reconstruction itself (the second step of the pipeline).

The second step of the pipeline concerns the reconstruction itself, which is, producing a 3D model of a target surface. The input for this step is the point cloud produced in the first step. This study considers unstructured point clouds. Thus, the only information available for the reconstruction are the 3D coordinates of the points sampled from the object surface. No information about the connectivity among the points, which depends on how the object was digitalized, is required by the methods presented here. Thus, these methods do not depend on how the point cloud was acquired. This step outputs a model of the target object surface represented by a polygonal mesh, more specifically a mesh of triangles, because a triangle is the simplest polygon, consequently it is easier to draw a triangle on the computer screen than any other polygon. Section 2.1 presents the polygonal mesh representation.

The last step of the pipeline is mesh simplification, that is, producing a simpler mesh (a mesh with less detail) from the mesh output in the second step. Some applications require meshes representing the shape of an object with less detail instead of quite dense meshes which a computer screen is slow to render. For this situation, a mesh simplification step is necessary.

Some desirable features of the reconstruction methods are: (a) To produce meshes of different resolutions so that the simplification step is not necessary; (b) To produce meshes with regular polygons, which in the case of triangles must be approximately equilateral; (c) To produce two-dimensional manifold meshes (2-manifolds); (d) To produce meshes that are a *Delaunay* Tesselation. A brief explanation about two-dimensional manifold and about the *Delaunay* Tesselation are presented in Sections 2.2 and 2.3 respectively.

2.1. Polygonal Meshes

Polygonal mesh representation is a particular case of boundary representation (b-rep), in which the faces delimiting the boundary of the solid are polygons, that is, plane figures bounded by a closed path, comprising a finite sequence of straight line segments (edges). A triangle is the simplest polygon, having only three edges. Most modeling and real-time

rendering software use the representation of triangular faces, because this requires less memory, less rendering time, and it adapts to any kind of contour. A surface representation in the form of a mesh of triangles consisting of a set of vertices, edges and triangular faces as illustrated in Fig. 1.

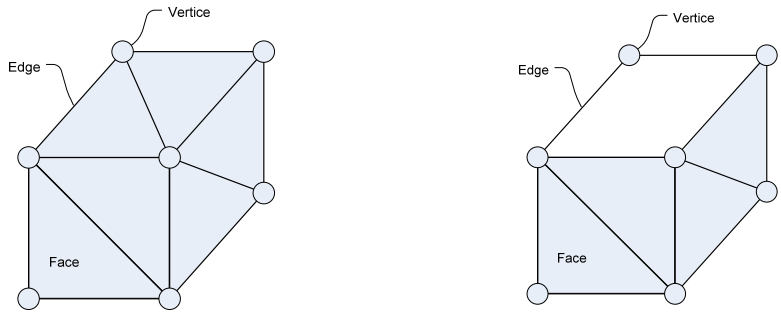


Fig. 1. Object surface representation using a mesh of triangles. Note that the two upper faces are missing in (b), for this reason the box in (b) seems to be open.

The boundary representation (b-rep) is widely used as a solid modelling technique. However, many b-rep systems support only solids the *boundaries* of which are *2-manifolds*. The next Section gives a brief explanation of two-dimensional manifolds.

2.2. Two-Manifolds

A two-dimensional manifold (2-manifold) is a topological space the points of which all have a neighborhood homeomorphic to an open disk (Mäntylä, 1988). In a 2-manifold mesh an edge is regular if it has exactly two coincident faces, and a vertex is regular if it has the same number of edges and faces (Barhak, 2002). If the surface has boundaries, then the mesh also has boundary edges and boundary vertices. The boundary edges have only one coincident face, and at the boundary vertices the number of coincident faces is one less than the number of emanating edges (Barhak, 2002). Thus, in a polygonal mesh, at most two faces can share the same edge. If more than two faces share an edge, the resulting mesh is not a two-dimensional manifold.

2.3. Voronoi Diagrams and Delaunay Triangulations

The surface reconstruction problem boils down to creating a mesh establishing its vertices and the connections between them forming triangular faces. For this reason, a geometric construction defining a triangle as the simplex building block is a suitable tool. Many studies (Amenta *et al.*, 2001), (Edelsbrunner *et al.*, 1983) have shown that *Delaunay* triangulations and its dual graph, the Voronoi Diagram, are well suited for surface reconstruction. The *Delaunay* triangulation presents the property of maximizing the minimum angle of the triangles in the mesh, thus avoiding skinny triangles. The reasons why skinny triangles should be avoided are explained in (de Berg *et al.*, 2008). In this section we give a brief introduction to Voronoi diagrams and *Delaunay* triangulation. More details

about these constructions can be found in (de Berg *et al.*, 2008) and (Aurenhammer & Klein, 2000).

Given a space $\mathbb{R}^n$ and a set S of nodes together with a notion of the influence that a node $s_i \in S$ with weight $\boldsymbol{\omega}_{s_i} \in \mathbb{R}^n$ exerts on a point $\boldsymbol{\omega}_k \in \mathbb{R}^n$, then the Voronoi region of s_i consists of all points $\boldsymbol{\omega}_k$ for which the influence of s_i is the strongest, over all $s_j \in S, j \neq i$ (Aurenhammer & Klein, 2000). Therefore, the Voronoi region of a node s_i in a map $S \subset \mathbb{R}^3$ is formally defined by (1) and the Voronoi diagram of the map S, denoted by $V(S)$, is a cell decomposition of $\mathbb{R}^3$ in convex polyhedrons. Each Voronoi cell around a node s_i contains all points of $\mathbb{R}^3$ that are closer to s_i than to any other node s_j. The complete diagram of the nodes and their Voronoi regions is called Voronoi diagram, as instanced in Fig. 2 (a).

$$VR(s_i) = \boldsymbol{\xi} \in \mathbb{R}^3 \mid \left\|\boldsymbol{\omega}_{s_i} - \boldsymbol{\xi}\right\| \leq \left\|\boldsymbol{\omega}_{s_j} - \boldsymbol{\xi}\right\| \; \forall s_j \in S, i \neq j \tag{1}$$

A *Delaunay* Tessellation of S, $DT(S)$, is obtained by connecting any two nodes s_i and s_j of S for which a circle C exists that passes through s_i and s_j and does not contain any other node in its interior or boundary. And if there are not four cocircular nodes of S, then $DT(S)$ - the dual of the Voronoi diagram $V(S)$ - is a triangulation of S, called the *Delaunay* Triangulation (Aurenhammer & Klein, 2000).

It is known that there is a unique *Delaunay* triangulation for S, if S is a set of points in a general position; that is, no three points are on the same line and no four are on the same circle, for a two dimensional set of points. For a set of points on the same line there is no *Delaunay* triangulation (in fact, the notion of triangulation is undefined for this case). For 4 points on the same circle (e.g., the vertices of a rectangle) the *Delaunay* triangulation is not unique: clearly, the two possible triangulations that split the quadrangle into two triangles satisfy the *Delaunay* condition. For a set of points S in 3d space, a *Delaunay* triangulation is such that every tetrahedron in $DT(S)$ has an empty circum-sphere. A set of points in n-dimensional Euclidean space is in a general position if no n + 1 points are on the same hyperplane and no n + 2 points are on the same hyper-sphere.

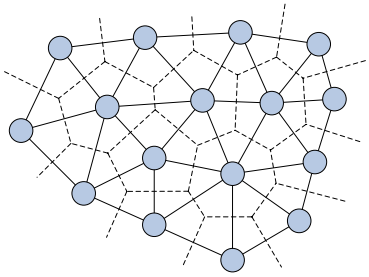


Fig. 2. Voronoi diagram and *Delaunay* triangulation. The dotted lines form the Voronoi diagram and the solid lines forms a *Delaunay* triangulation.

3. Using Self-Organizing Maps for Surface Reconstruction.

When self-organizing maps are employed for reconstruction, the mesh vertices correspond to the output nodes of the map. The weight vector of the nodes determines the positions of

the vertices in the mesh, whereas the connections between the nodes correspond to the edges of the mesh.

Self-Organizing Maps, a kind of Artificial Neural Network with unsupervised learning, consist of an input and an output layer. Each input node is connected to every output node. The output nodes may be connected to each other. When self-organizing maps are employed for reconstruction, the input message and the nodes in the output layer have a weight vector representing a 3D coordinate $[x, y, z]^T$. The output layer represents the polygonal mesh being reconstructed: The mesh vertices correspond to the output nodes of the map; the weight vector of the output nodes determines the positions of the vertices in the mesh; the connections between the output nodes correspond to the edges of the mesh. For this reason, the words node and vertex, connection and edge, are used interchangeably.

The faces of a polygonal mesh, however, do not have a direct representative in the Self-Organizing Map. If the connections between the nodes are fixed, as in the Self-Organizing Map (SOM) originally proposed in (Kohonen, 1998), then the faces can be established in advance. The faces of a triangular mesh can also be represented by the basic building block of a SOM variant called Growing Cell Structures (GCS) (Fritzke, 1994). Other SOM variants, such as the Topology Representing Network (TRN) (Martinez & Schulten, 1994) and Growing Neural Gas (GNG) (Fritzke, 1995), do not have a direct representative for the faces of a polygonal mesh.

In the original SOM (Kohonen, 1998) the connections among nodes are defined by their position in the lattice, so that a planar mesh is defined. The learning phase deforms and adjusts this initial mesh to couple with the objective surface. So, at the end of this phase, the distribution of the weight vectors of the nodes in the lattice comes with the surface folds. To obtain this, an important issue is the way in which the samples are presented to the SOM network. The learning phase produces a map that represents the probability density function (pdf) of the input data. For a 2D surface immersed into 3D space, this is one for all points on the surface and zero otherwise. However, this is not sufficient because the scanning phase could produce a point cloud where the occurrence of each sample is not equal. Some surface parts could be over sampled resulting in a point cloud in which some samples are repetitive, or which has a lot of very similar samples. Therefore a pre-processing step is need. Moreover, to avoid the effects of over-fitting nodes, the samples should be presented to the Network in a random sequence. Another relevant issue is the size of the lattice which has to be specified in advance. This implicitly specifies the level of detail preserved by the reconstructed surface model. To specify a suitable size, one should consider the total area of the surface and the size of the triangles in the mesh in order to preserve the richness of the details as required. Despite all these achievements, SOM has displayed difficulties in reconstructing concave regions. That is, to say, after the learning process has taken place, some vertices or triangles may be unstable, and dangle among regions densely populated by the data.

As in SOM the TRN model begin with a pre-defined number of nodes. However, the TRN initial map does not define an initial mesh as in SOM. TRN is a combination of Neural Gas (NG) (Martinez & Schulten, 1991) and Competitive Hebbian Learning (CHL) (Martinez, 1993). The former distributes the nodes according to the probability density function of input data, while the latter builds a mesh connecting these nodes in a fashion that reproduces the topology of the objective surface. As the connections between nodes are

defined later, after the nodes have been placed next to their final position, the learning process is improved.

A solution for the limitation represented by size of the of network, which in SOM and TRN needs to be specified in advance, is found in Growing models such as (GCS) and (GNG). These models build maps that grow incrementally as the samples are presented. The GCS generates maps consisting only of a basic building blocks type, triangles for surface reconstruction. As opposed to GCS, a map generated by GNG may have nodes with different connectivity and the topology may have different dimensionalities in different parts of the map. Like the TRN, the GNG is able to learn the topology of input data through Competitive Hebbian Learning. Hence, GNG can be seen as a GCS variant without its topological restrictions or as a growing version of TRN.

The Neural Mesh (NM) (Ivrissimtzis *et al.*, 2004a) is a growing model that starts from an initial mesh forming a tetrahedron and grows or diminishes by splitting the more active vertex or removing the least active vertex. This is a probabilistic algorithm in which the output is dependent on the presentation sequence of the samples. Thus, an ensemble combining several reconstructions into a single one, may be necessary in order to reach a quality reconstruction. The most undesirable NM limitation is the absence of any guarantee that a sufficiently good point cloud will result in a correct and faithful reconstruction (Ivrissimtzis *et al.*, 2004b). In particular, the algorithm has difficulties in distinguishing between two close sheets.

Growing Self-Reconstruction Meshes (GSRM) (do Rêgo *et al.*, 2007, 2009) extend Growing Neural Gas (Fritzke, 1995) by including the concept of triangular faces in the learning algorithm and additional conditions in order to include and remove connections, so that it can produce a triangular 2-manifold mesh representation of a target object given an unstructured point cloud of its surface. The main modifications concern Competitive Hebbian Learning, the vertex insertion operation and the edge removal mechanism. GSRM is able to reproduce the shape of the original object by learning the geometry and topology of the surface represented in the point cloud. Also, GSRM generates meshes with different resolutions during the learning process.

The Growing Self-Organizing Surface Map (GSOSM) (DalleMole & Araújo, 2008a, 2008b, 2009) aims to produce a mesh of equilateral triangles faithful to the object surface. The mesh is constructed in a growing fashion by a learning process starting from a cloud of points sampled on the object surface. The learning process inserts nodes and connections to represent the vertices and the edges of triangles of the mesh aimed at. The insertion of nodes considers an empty receptive field whereas the triangles edges are learned using a new learning rule called Competitive Connection Hebbian Learning (CCHL). This rule considers the three nodes closest to each input sample as candidate to form a triangle. Thus, for each input sample, CCHL inserts only one new edge. The approximate equilaterally of triangles is obtained as a consequence of the adaptation steps of the learning process. The final mesh is generally a two-manifold mesh without holes.

The last two models are novel and attempt to overcome several of the limitations of their predecessors, for example, dependence on cycle counters and error accumulators (GSOSM); topology learning and multi-resolution meshes (GSRM). These models are presented in detail below.

4. Growing Self-Organizing Meshes (GSRM)

Growing Self-Organizing Meshes is a surface reconstruction method based on growing self-organizing maps, which learns both the geometry and the topology of the input data set. GSRM produces 2-manifold meshes that approximate very well the shape of an object, including its concave regions and holes, if any. The maps produced with GSRM grow incrementally producing meshes of different resolutions, according to different application needs. Other important features of GSRM are that the triangular faces of the meshes are approximately equilateral and most of the edges satisfy the local criterion of the *Delaunay* triangulation of piecewise flat surfaces (Bobenko & Springborn, 2007), (Fisher *et al.*, 2007).
The GSRM has GNG as its starting point and modifies the following steps: node insertion, removal of nodes and faces, creation of nodes and faces. The modification aims at producing 2-manifold meshes of triangles, since standard GNG maps do not represent the faces of a polygonal mesh. The steps mentioned above are explained in Section 4.3 to 4.5. Section 4.1 presents the steps of the GSRM learning algorithm. Section 4.2 discusses the parameters of the algorithm. Section 4.6 presents the topological learning step performed to complete the topological learning after the desired mesh resolution is reached. Section 4.7 presents the post-processing step necessary to complete the triangulation of the mesh output by GSRM. Some experimental results of the GSRM reconstruction are presented in Section 4.8.

4.1. The GSRM Algorithm

The GSRM learning algorithm is briefly described below:

1. Initialize the map A with three nodes with weight vectors randomly chosen from a point cloud P.
2. Present a sample $\boldsymbol{\xi}$, randomly chosen from P.
3. Find the two nodes (s_1 and s_2) of the map that are nearest to $\boldsymbol{\xi}$ according to the Euclidean distance.
4. Create connections and faces according to the ECHL (Section 4.3). Sometimes connections are reinforced instead of created. In this case, other edges are checked against a condition for edge removal based on the Thales Sphere concept (Section 4.4).
5. Update the error counter of node s_1:

$$\Delta E_{s_1} = \left\| \boldsymbol{\xi} - \boldsymbol{\omega}_{s_1} \right\| \tag{2}$$

   where, $\boldsymbol{\omega}_{s_1}$ is the weight vector of node s_1.
6. Adapt the weight vector of the winner node s_1 and its neighbors.

$$\Delta \boldsymbol{\omega}_{s_1} = \varepsilon_b \left\| \boldsymbol{\xi} - \boldsymbol{\omega}_{s_1} \right\| \tag{3}$$

$$\Delta \boldsymbol{\omega}_{s_n} = \varepsilon_n \left\| \boldsymbol{\xi} - \boldsymbol{\omega}_{s_n} \right\|, s_n \in N(s_1) \tag{4}$$

   where $\mathrm{N}(\mathrm{s}_1)$ is the neighborhood of s_1.
7. Update the age of all edges e emanating from s_1.

$$age_e = age_e + 1 \tag{5}$$

8. Remove the faces coincident to an old edge e ($age_e > age_{max}$) and remove this edge (Section 4.4).
9. If the number of samples presented so far is greater than λ , insert a new node in the map (Section 4.5).
10. Decrease the error variables of all nodes:

$$\Delta E_s = -\beta E_s \, , \forall s \in A \tag{6}$$

11. If the map has achieved the desired resolution, complete the topological learning (Section 4.6).

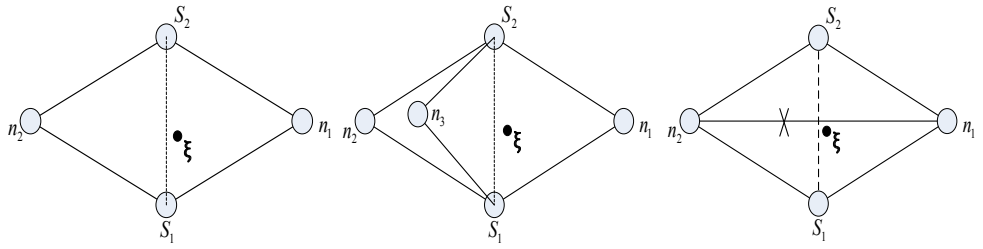
4.2. Parameters

The GSRM learning algorithm relies on six parameters: ε_b and ε_n – learning rate of the winner node and its neighbors, respectively ($\varepsilon_b > \varepsilon_n$); λ – the frequency at which a new node is inserted; α – the error reduction rate of the nodes that are neighbors of a node that has just been inserted; β – the error reduction rate that aims at stressing the impact of recently accumulated errors ($\beta < \alpha$); age_{max} – the maximum age at which an edge can be removed.

4.3. Creation of Edges and Faces

The mechanism that GSRM uses for the creation of edges and faces is an extension of standard Competitive Hebbian Learning (CHL) that we call Extended Competitive Hebbian Learning (ECHL). The goal is to extend the CHL so that instead of defining only edges, it can also define the faces of a triangular mesh representation. Moreover, ECHL avoids the creation of overlapping edges and non-manifold meshes. The ECHL algorithm is described below:

1. Present a randomly chosen sample $\boldsymbol{\xi}$.
2. Select the nodes of the map s_1 and s_2 that are nearest to $\boldsymbol{\xi}$ according to the Euclidian distance. See Fig. 3 (a) for an example.
3. If s_1 and s_2 are not connected by an edge, create such an edge, since s_1 and s_2 do not have more than two neighbors in common. This condition avoids more than two faces incident on the same edge, so that the mesh remains a 2-manifold Fig. 3 (b) illustrates what would happen without this condition).
   3.1. If s_1 and s_2 have two common neighbors (n_1 and n_2) connected by an edge, this edge is removed together with its coincident faces. This step avoids overlapping edges. In the example of Fig. 3 (b) the edge between n_1 and n_2 would be removed, so the it does not overlap that new edge e_{s_1,s_2}.
   3.2. Create new face(s) comprising s_1, s_2 and each of their common neighbors, if any. See Fig. 3 (c) for an example.
4. Otherwise
   4.1. Reinforce the existing edge e_{s_1,s_2} by setting its age to zero. This reduces the possibility of the edge connecting s_1 and s_2 being removed.
   4.2. Check if some edge emanating from s_1 against the edge removal condition based on the Thales sphere concept (Section 4.4).



(a) (b) (c)

Fig. 3. Extended Competitive Hebbian Learning: a) The nearest nodes s_1 and s_2 for the sample $\boldsymbol{\xi}$ and its common neighbors n_1 and n_2; b) The manifoldness of the mesh is broken, if the connection between the nodes s_1 and s_2 is created; c) The sample $\boldsymbol{\xi}$ induces the establishment of a connection between the nodes s_1 and s_2 crossing the connection between the nodes n_1 and n_2 which is removed.

4.4. Removal of Edges and Faces

GSRM uses two schemes to decide whether an edge is obsolete and should be removed. The first is the edge ageing scheme used in the standard GNG. The second is based on the Thales sphere concept, and it is also used in GSOSM. The edge ageing scheme removes an edge when its age becomes greater than a given threshold (age_{max}). The age of an edge is zero when it is created, and increases as described in step 7 of the algorithm presented in Section 4.1. The age of an edge can be reset, as described in step 4.1. of the algorithm presented in Section 4.3.

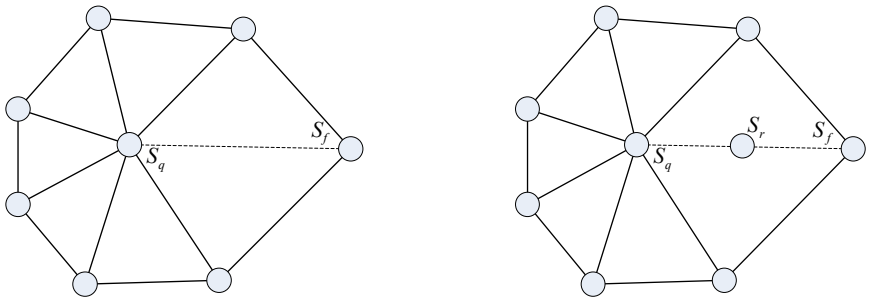
As the position of the nodes change during the learning process, two connected nodes may become distant from each other, implying that the mesh has long edges. These long edges should be removed; otherwise, the algorithm fails to meet the goal of having triangular faces which are approximately equilateral. To remove these long edges an edge removal scheme based on the ideas of (Jockusch & Ritter, 1993) is applied. According to this scheme, whenever a connection to be generated by a sample presentation already exists, the edges connecting the winner node s_1 to each of their neighbors s_k are considered candidates for removal. A connection between s_1 and s_k is removed when the second winner node s_2 is inside the Thales sphere with a diameter of $\left\|\boldsymbol{\omega}_{s_1} - \boldsymbol{\omega}_{s_k}\right\|$, where $\boldsymbol{\omega}_{s_1}$ and $\boldsymbol{\omega}_{s_k}$ are respectively the weight vectors of the nodes s_1 and s_k. To verify this condition the angle between the vectors $\mathbf{v} = \boldsymbol{\omega}_{s_1} - \boldsymbol{\omega}_{s_k}$ and $\mathbf{u} = \boldsymbol{\omega}_{s_k} - \boldsymbol{\omega}_{s_1}$ is determined. If this angle is greater than $\pi/2$, then the condition is satisfied and the edge between s_1 and s_k is removed (Fig. 11 (a) illustrates this situation). Otherwise, the condition is false.

If an edge must be removed, either because of its age or because of the Thales sphere based condition, the removal of this edge is performed after the removal of every incident face. After the removal of the edge, the vertices of the newly removed edge may not have an edge emanating from it, and must be removed as well.

4.5. Node Insertion

A new node is always inserted in the map after a number λ of adaptation steps. The new node s_k is inserted between the node s_q with the highest error counter and its neighbor s_f with highest error counter, this means that the weight vector of s_k is initialized as: $\boldsymbol{\omega}_{s_k} =$

$0.5(\boldsymbol{\omega}_{s_q} + \boldsymbol{\omega}_{s_f})$. The edge connecting s_q to s_f is removed together with its coincident faces. New edges connecting s_k to s_q and s_k to s_f are created. Fig. 4 illustrates this process: (a) it presents a mesh before insertion of the node, and (b) it presents the same mesh after the node has been inserted. The error counters of nodes s_q and s_f are decreased according to parameter β ($\Delta E_s = -\beta E_s$). Finally, the error counter of the new node is interpolated from the error counter of s_q and s_f ($E_{s_r} = 0.5(E_{s_q} + E_{s_f})$).



(a) (b)

Fig. 4. Example of a mesh before (a) and after (b) a new node insertion.

4.6. Topology Learning

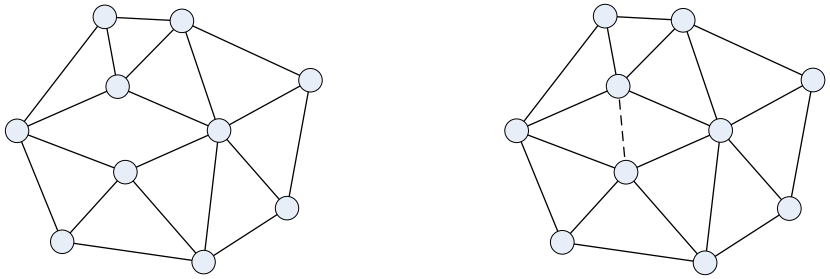
The GSRM learning algorithm behaves non-deterministically. Some edges needed for the reconstructed mesh may not have been created when the learning process finishes, because no sample that would trigger these edges was presented at the appropriate moment of the learning process. Also, some long edges may not have been removed since the Thales sphere based condition to verify if an edge is long enough and must be removed is not checked at each iteration, but only periodically. Thus, to complete the topological learning, a deterministic topological learning step is performed after the main learning processing is finished. At this moment, the vertices no longer change their position (weight vectors). First, every edge is checked against the Thales sphere based condition to verify if any edge is too long and must therefore be removed. Secondly, every sample is presented so that, all the necessary edges, that is, connections between nodes, are created. Note that since the vertices are no longer changing their positions, no edge becomes unseful during this final topological learning step.

4.7. Post Processing Step

After the topology learning step, some approximately regular polygons remain untriangulated because the two winner nodes for any sample internal to these polygons are already connected (see Fig. 8). In our experiments, the vast majority of the untriangulated polygons had four or five vertices, a few of them had six vertices, and none of them had more than six vertices. The post processing step aims at triangulating these polygons.

As to the quadrilaterals, triangulation is performed by creating one of their edges and replacing the quadrilateral with two triangles. See Fig. 5 for an example. The diagonal chosen is the one that makes the sum of the opposite angles in the adjacent triangles less than π. This choice leads to edges that satisfy the local *Delaunay* condition for piecewise flat

surfaces (Bobenko & Springborn, 2007), (Fisher *et al.*, 2007). The triangulation of polygons with more than four vertices is performed by inserting a new vertex at the geometric center of the polygon and connecting this new vertex to each of the vertices of the polygons. See Fig. 6 for an example.



(a) (b)

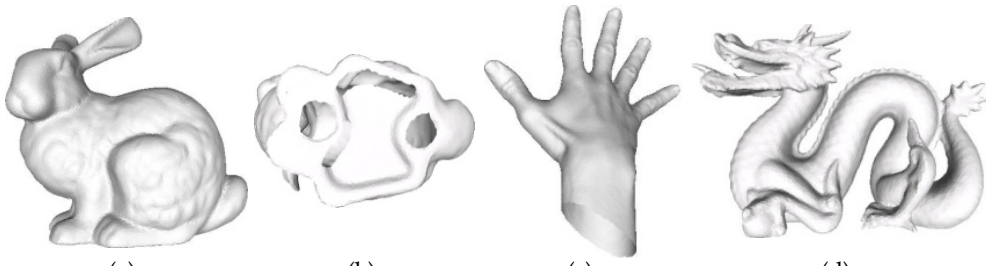
Fig. 5. Quadrilateral before (a) and after (b) triangulation.

(a) (b)

Fig. 6. Pentagon before (a) and after (b) triangulation.

4.8. Experimental Results

This Section presents some reconstructions produced by GSRM. Fig. 7 shows pictures of three synthetic objects: Bunny and Dragon available at the Stanford Repository (graphics.stanford.edu/data/3Dscanrep), and Hand, donated by Ioannis Ivrissimtzis (Ivrissimtzis *et al.*, 2004). All the reconstructions presented in this Section have about 20,000 vertices. Note that the shape of the objects is represented in the GSRM reconstructions as are their concave regions and holes (Fig. 7 (b)).



(a) (b) (c) (d)

Fig. 7. Example of two reconstructions produced with GSRM: a) and b) are respectively a lateral view and a view of the underside of reconstructed version of the Bunny model; c) and d) lateral views of reconstructed versions of the Hand and the Dragon models.

The numerical results concern: distance from the target object, valence distribution, polygon conformity and the number of *Delaunay* edges. The Hausdorff distance between the target and the reconstructed objects was calculated with the Metro tool (Cignoni *et al.*, 1998). Valence distribution refers to the number of neighbors of the mesh vertices. Valences must be distributed as evenly as possible, i.e., ideally all vertices should have the same valence. Polygon conformity R(P) is measured as the ratio between the smallest and the largest distance of its vertices (s) to the polygon baricenter (bp). The *Delaunay* condition of piecewise flat surfaces presented in (Bobenko & Springborn, 2007) has been applied to verify the number of valid *Delunay* edges of the GSRM reconstructions. Table 1 shows the following measurements: the Hausdorff Distance, the average polygon conformity and the percentage of *Delaunay* edges of the meshes reconstructed with GSRM. The average percentage of the valences presented by the vertices of the meshes reconstructed with GSRM is shown in Table 2.

Model	Hausdorff Distance	Polygon Conformity (avg)	Delaunay Edges (%)
Bunny	0.001510	0.688785	99.86
Dragon	0.023644	0.684732	99.81
Hand	0.001364	0.689153	99.83

Table 1. The Hausdorff distance, the average polygon conformity and the percentage of *Delaunay* edges of GRSM reconstructions.

Model	Valence Distribution (%)					
	4	5	6	7	8	others
Bunny	4.32	25.76	42.49	21.87	4.76	0.8
Dragon	5.55	25.66	41.10	22.15	5.38	1.15
Hand	4.27	25.7	42.45	25.03	4.88	0.66

Table 2. Valence distribution of the vertices of GSRM reconstructions.

5. Growing Self-Organizing Surface Map (GSOSM).

This model overcomes several restrictions of its predecessors, the mapping process is incremental and there are no dependences on error accumulators or cycle counters. Moreover, the model is able to handle correlated samples and disjoint close surface parts and to recover a surface representation from multiple subsets of an entire point cloud blending the surface parts automatically, in response to the presentation of data. However, GSOSM is not suitable for producing surface representation with different levels of detail. The GSOSM learns the surface starting from an empty map and inserts nodes and connections between them, thus producing a mesh of equilateral triangles. This mesh is the surface representation and, therefore, the preservation of richness of detail is dependent only on the length of the edges of the triangles.

5.1. The GSOSM Algorithm

The GSOSM algorithm comprises nine steps:

1. Parameter setup;
2. Presentation of a new sample;
3. Determination of the three closest nodes to the current sample;
4. Insertion of a new node;
5. Weight vector update;
6. Node collapse;
7. Determination of the connection that captures the current sample;
8. Insertion or substitution of connection;
9. Removal or swap of connections.

5.2. Parameters

The GSOSM has four parameters: e_{max} - defines the size of edges of triangles; α - the learning rate; θ_{min} - specifies the largest value allowed for an internal angle of a triangle; and β_{min} – represents a constraint to connection insertion and is employed to distinguish between parallel close sheets of surface.

5.3. The Nodes Insertion Operator

A GSOSM reconstruction starts with an empty map S and the first node is inserted with the first input signal. A spherical receptive field with a radius of e_{max} is considered for each node and a new node is inserted only if the input signal $\boldsymbol{\xi}$ is out of any receptive field. When a new node s_{new} is inserted its weight vector $\boldsymbol{\omega}_{s_{new}}$ is equal to the current sample $\boldsymbol{\xi}$, and the node does not have any initial connection.

5.4. The Adaptation Operator

The objective of the adaptation is to adjust the map to the surface and consists of two distinct sub-operations. The first is the original SOM adaptation step (7) and moves the winner node towards the input signal $\boldsymbol{\xi}$ according to the learning rate α. This step is applied only if the three nodes closest to the input signal are not connected and do not form a triangle. The second sub-operation (8) rotates the triangle about the axis defined by the second and third closest nodes, towards the input signal.

$$\boldsymbol{\omega}_{s_0} = \boldsymbol{\omega}_{s_0} + \alpha[\boldsymbol{\xi} - \boldsymbol{\omega}_{s_0}] \tag{7}$$

$$\boldsymbol{\omega}_{s_0} = \boldsymbol{\omega}_{s_0} + \alpha\, \mathbf{P}_n\, d(P, \boldsymbol{\xi}) \tag{8}$$

where: $\mathbf{P}_n$ is the normal vector of plane P formed by the three closest nodes for $\boldsymbol{\xi}$, and $d(P, \boldsymbol{\xi})$ is the distance of $\boldsymbol{\xi}$ to the plane P.
The original SOM adaptation operator applied to correlated and repetitive samples pulls the winner node towards the second closest node undoing the equilateral triangles. However, this does not occur when the schema above is employed. Therefore, the GSOSM is able to handle repetitive samples.

5.5. The Merging Operator

The operator of insertion of new nodes ensures a minimal distance, greater than e_{max}, between any two nodes. However, all nodes are subject to being repositioned by the adaptation operator. As a consequence, a repositioned node can invade the receptive field of another node. Thus, the GSOSM applies the merging operator and transforms these nodes into one that is positioned midway between them.

5.6. The CCHL Learning Rule

The GSOSM introduces a very nice learning rule named Competitive Connection Hebbian Learning (CCHL). This new learning rule was proposed to substitute the CHL (Martinez & Schulten, 1991) as the rule to learn the node-neighborhood relationships among nodes. The new rule overcomes the failure of CHL to complete a triangulation inside polyhedrons such as a quadrilateral as illustrated in the Fig. 8.

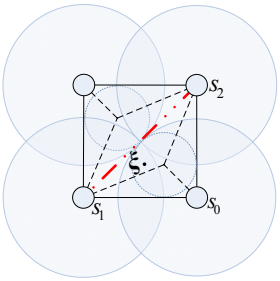
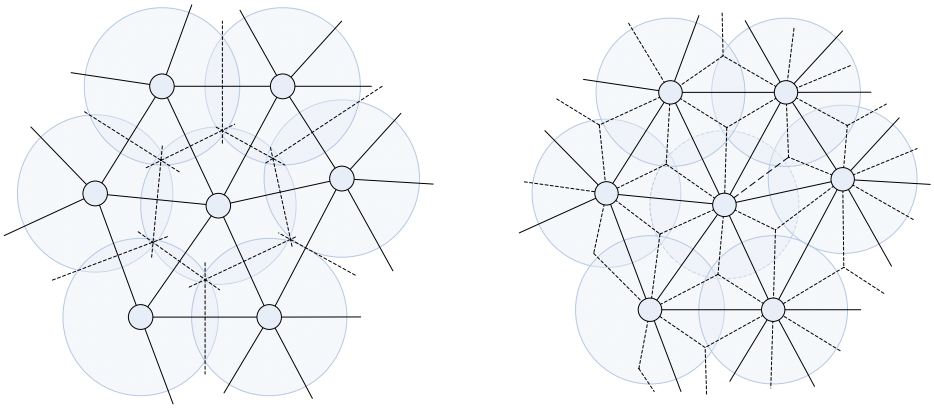


Fig. 8. A case where CHL Learning rule fail to complete the triangulation.

The CHL rule performs a competition among all nodes and the two nodes closest to the current sample are selected to be connected. Now consider four interconnected nodes forming a quadrilateral without any diagonal and an input signal $\boldsymbol{\xi}$ within it (Fig. 8). In this configuration, the most activated nodes are s_0 and s_1. Therefore, CHL would insert a connection between them. However, this is an existing link and to complete the local triangulation a new connection should be inserted between s_1 and s_2, to form two triangles. The pentagon and hexagon are similar cases.

The basis of the CCHL learning rule is Voronoi diagrams, that is a cell decomposition of the space into convex polyhedrons. Similar to the Voronoi cell of a node (Fig. 9 (a)), each Voronoi cell around a connection $c_{s_i s_j}$ contains all points that are closer to $c_{s_i s_j}$ than to any other $c_{s_k s_l}$. A sample of a diagram of connection Voronoi cells $V^c(S)$ in the $\mathbb{R}^2$ space is shown in Fig. 9 (b). If all triangles are equilateral, then $V^c(S)$ is equal to the second order Voronoi diagram.



(a) (b)

Fig. 9. $\mathbb{R}^2$ Voronoi Diagrams: a) Node receptive field Voronoi cells; b) Connection Voronoi cells.

In CCHL, the competition for the connection insertion has two distinct phases. The first one determines the three nodes closest to the current sample. The second phase considers the three nodes as vertices of an imaginary triangle and the closest edge to $\boldsymbol{\xi}$ is the connection that should be inserted, if it does not already exist. So, the CCHL can be stated as: given an input signal $\boldsymbol{\xi}$ and its three closest nodes $\{ s_0, s_1, s_2\}$, the pair to be connected is that forming the edge closest to $\boldsymbol{\xi}$.

Return to Fig. 8, notice the stippled lines denoting the Connection Voronoi Cells and the winner connection linking the nodes s_1 and s_2. This connection split the lozenge into two triangles, thus completing the local triangulation. Nonetheless, CHL would recommend inserting the connection between nodes $s_0$and s_1, thus not completing the triangulation.

5.7. The Connection Insertion Operator

In the GSOSM, the mesh that represents the surface is implicitly defined by the nodes and its connections. The CCHL rule is used to decide the connection $c_{new}(s_i, s_j)$ that should be inserted. However, the GSOSM imposes three other conditions that need to be satisfied before creating a connection.

1) The coefficient of correlation β (9) between the activation of s_i and s_j needs to satisfy (10). This condition is used to verify the continuity of the input space between the nodes s_i and s_j.

$$\beta = \frac{(\boldsymbol{\xi} - \boldsymbol{\omega}_{s_i}).(\boldsymbol{\omega}_{s_j} - \boldsymbol{\omega}_{s_i})}{\left\|\boldsymbol{\omega}_{s_j} - \boldsymbol{\omega}_{s_i}\right\|^2} \quad , \quad \left\|\boldsymbol{\xi} - \boldsymbol{\omega}_{s_i}\right\| \leq \left\|\boldsymbol{\xi} - \boldsymbol{\omega}_{s_j}\right\| \tag{9}$$

$$\beta_{min} \leq \beta \leq 0.5 \qquad \beta_{min} \in [0, 0.5] \tag{10}$$

As the projection of ξ approximates to the connection midpoint (Fig. 10) the probability of inserting a connection crossing an empty area between two close sheets vanishes.

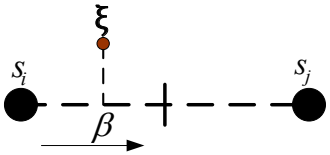


Fig. 10. Visualization of coefficient β as a projection of the input signal over the winner connection.

2) The coefficient of similarity θ (11) between s_i and s_j with respect to node s_k is greater than or equal to θ_{min}. Geometrically, θ represents the cosine of the angle between vectors $\mathbf{v} = \boldsymbol{\omega}_{s_i} - \boldsymbol{\omega}_{s_k}$ and $\mathbf{u} = \boldsymbol{\omega}_{s_j} - \boldsymbol{\omega}_{s_k}$, with s_i and s_j being the nodes in the extremities of the winner connection and s_k the other closest node. This condition assures the formation of triangles that are approximately equilateral.

$$\theta = \frac{\boldsymbol{\omega}_{s_i} - \boldsymbol{\omega}_{s_k}}{\left\|\boldsymbol{\omega}_{s_i} - \boldsymbol{\omega}_{s_k}\right\|} \cdot \frac{\boldsymbol{\omega}_{s_j} - \boldsymbol{\omega}_{s_k}}{\left\|\boldsymbol{\omega}_{s_j} - \boldsymbol{\omega}_{s_k}\right\|} \tag{11}$$

3) The winner connection $c_{win}(s_i, s_j)$ does not cross the Voronoi region of any other connection, thus avoiding the creation of overlapping triangle faces. The crossings can be found using the CCHL to verify if the sample midpoint of c_{win} is within a Voronoi region of another connection $c_{s_k s_t}, t \neq i, j$. If an overlap is detected the value of the coefficient ϕ of each overlapping connection is used to decide which should be removed. The coefficient ϕ (12) is a measure of the adherence of the connection to the objective surface and is obtained as follows:

$$\phi = \left(1 + |\beta - 0.5| + \frac{d(c_{s_i s_j}, \boldsymbol{\xi})}{\left\|\boldsymbol{\omega}_{s_i} - \boldsymbol{\omega}_{s_j}\right\|}\right)^{-1} \tag{12}$$

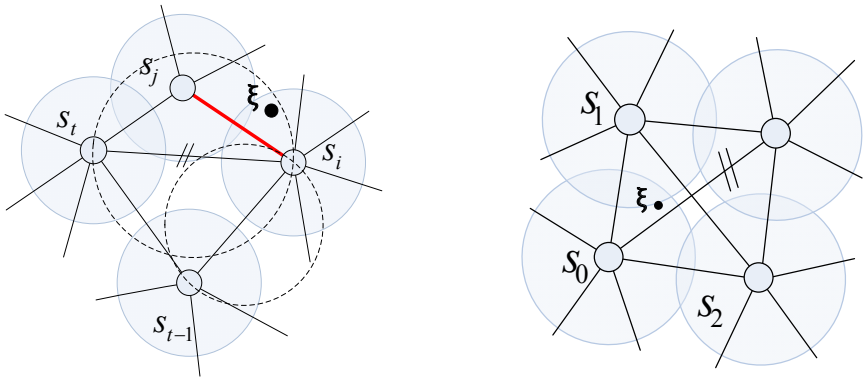
with $$d\left(c_{s_i s_j}, \boldsymbol{\xi}\right) = \left\|\left[\boldsymbol{\omega}_{s_i} + \beta\left[\boldsymbol{\omega}_{s_j} - \boldsymbol{\omega}_{s_i}\right]\right] - \boldsymbol{\xi}\right\|$$

where: $\boldsymbol{\omega}_{s_i}$ and $\boldsymbol{\omega}_{s_j}$ are the weight vectors of the nodes at the extremities of the connection $c_{s_i s_j}$.

5.8. The Connection Removal Operator

The connection removal operator in GSOSM consists of two steps. The first removes or swaps connections that could produce triangles with an internal θ angle the cosine (13) of which is lower than the parameter θ_{min} as shown in Fig. 11 (a).

$$\frac{\boldsymbol{\omega}_{s_i} - \boldsymbol{\omega}_{s_j}}{\left\|\boldsymbol{\omega}_{s_i} - \boldsymbol{\omega}_{s_j}\right\|} \cdot \frac{\boldsymbol{\omega}_{s_t} - \boldsymbol{\omega}_{s_j}}{\left\|\boldsymbol{\omega}_{s_t} - \boldsymbol{\omega}_{s_j}\right\|} < \theta_{min} \tag{13}$$



(a) (b)

Fig. 11. The process of detection and removal or swap of connections: a) long connections; b) crossing connections forming overlapping triangles.

The second step is used to eliminate overlapping triangle faces. The removal of cross connections is carried out using the CCHL to verify if the midpoint of c_{win} is within another connection Voronoi region, as discussed earlier. If an intersection is detected (Fig. 11(b)), the operator removes the connection with the smallest ϕ value (12).

5.9. Experimental Results

In this Section, numerical and visual results of GSOSM reconstructions are reported. The models used come from the Stanford 3D Scanning Repository (graphics.stanford.edu/data /3Dscanrep) and the AIM@Shape Repository (www.aimatshape.net). The general quality of GSOSM reconstructions can be visually accessed in Fig. 12 which shows views of four reconstructed object models. Notice that GSOSM preserves the details of the original surface in the reconstructed version (Fig. 12 (a) and (d)) and the suavity of the curves of the surface (Fig. 12 (b)). Moreover, the ability of GSOSM in reproducing close sheets correctly is showed in the Fig. 12 (c). The equilaterality quality of the triangles in a mesh produced by GSOSM can be inspected visually at Fig. 13.

Table 3 shows measurements on the quality of the meshes produced by GSOSM. The columns under the caption "Valence" show that the meshes produced are very regular with each vertex being a centre of a hexagon. The next column shows that GSOSM is able to produce a mesh in which most of its edges are valid *Delaunay* edges. The last columns, under the caption "Equilaterality" show the measured quality of the triangles in terms of

their equilaterality. These values were calculated using the equilaterality quality measure defined by Rypl (2005), that is.

$$q = f\frac{A}{a^2 + b^2 + c^2} \tag{14}$$

where, A represents the area of the triangle, a, b and c are the lengths of its edges and $f = 4\sqrt{3}$ is a normalizing coefficient which justifies the quality of an equilateral triangle to one.

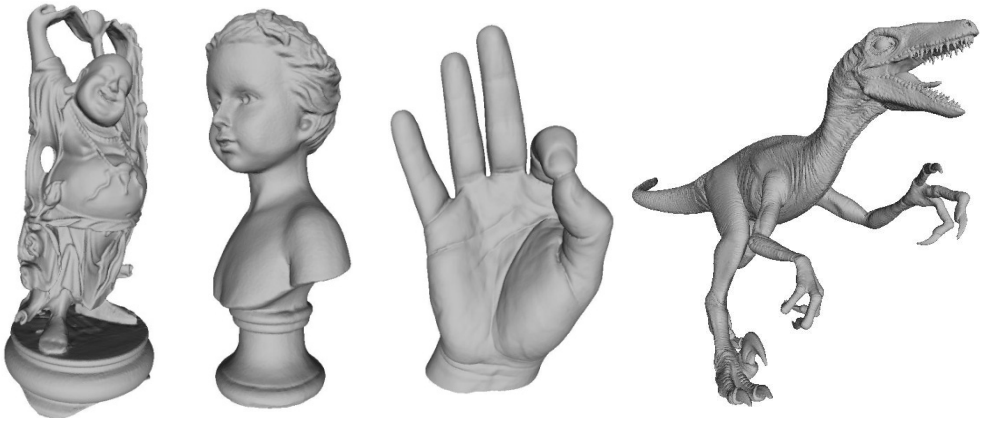


Fig. 12. Views of GSOSM object surface reconstructions: a) the Happy Buddha; b) the Bust; c) the Gips Hand; d) the Raptor;

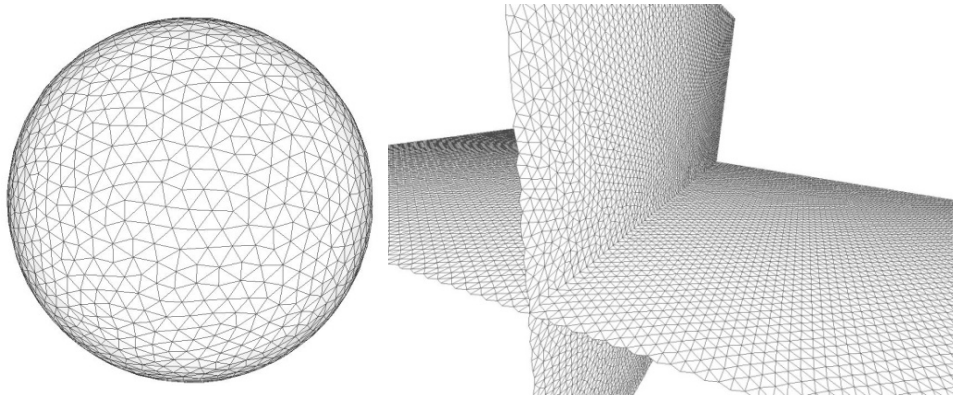


Fig. 13. Visual inspection of the equilaterality quality of the triangles: a) a sphere; b) a non 2-manifold surface.

Notice that the column "min" indicates the existence of some skinny triangles, this is because of the coefficient ϕ which forces the permanence of long connections if and when they are fitted to the objective surface fold. However, the following three columns show that high quality triangles are the great majority and that the skinny triangles are exceptions.

Object	Triangles						
	Valence		% invalid Delaunay Edges	Equilaterality (q)			
	avg.	std. dev.		min.	max.	avg.	std. dev.
Bunny	5.990	0.724	1.53	0.216	1.000	0.936	0.058
Asian Dragon	5.999	0.718	1.51	0.210	1.000	0.933	0.067
Dragon	5.999	0.766	2.50	0.181	1.000	0.931	0.073
Happy Buddha	5.999	0.759	2.30	0.208	1.000	0.933	0.071
Armadillo	5.999	0.772	2.66	0.232	1.000	0.930	0.074
Lucy	5.999	0.667	1.21	0.178	1.000	0.945	0.061
Thai Statue	5.998	0.712	1.62	0.160	1.000	0.940	0.065
Filigree	6.002	0.688	1.80	0.187	1.000	0.938	0.064
Bust	5.999	0.788	2.73	0.236	1.000	0.927	0.078
Hand	5.996	0.805	2,59	0.204	1.000	0.928	0.078
Gips Hand	5.995	0.770	2.34	0.219	1.000	0.933	0.070

Table 3. Statistics on triangles from GSOSM reconstructions.

The results reported in Table 4, were computed using the Metro tool (Cignoni *et al*., 1998) and are indicative of the fidelity of GSOSM reconstructions to the original object surfaces. In this table, $\mathcal{M}_1$ refers to the reconstructed mesh while $\mathcal{M}_2$ is the original mesh. The labels are: "max" for the maximum calculated value of the distance between the surfaces $\mathcal{M}_1$ and $\mathcal{M}_2$; "avg" for the mean distance; and "rms" for the root mean square. The column "H" is the Hausdorff distance, which is equal to the greatest maximum value.

Model	Metro results (x 10-3)						
	$\mathcal{M}_1 \rightarrow \mathcal{M}_2$			$\mathcal{M}_2 \rightarrow \mathcal{M}_1$			H
	max	avg	Rms	max	avg	rms	
Bunny	1.419	0.040	0.064	1.490	0.043	0.078	1.490
Dragon	1.348	0.018	0.031	1.807	0.023	0.119	1.807
Happy Buddha	1.389	0.024	0.042	1.255	0.034	0.146	1.389
Armadillo	0.870	0.025	0.037	0.925	0.026	0.040	0.870
Filigree	1.180	0.019	0.043	1.827	0.021	0.050	1.827
Bust	0.788	0.019	0.031	1.135	0.020	0.034	1.135
Gips Hand	0.646	0.022	0.033	1.392	0.025	0.038	1.392
Hand	0.909	0.011	0.017	1.588	0.011	0.021	1.588

Table 4. Metro measures of the reconstructions of models using GSOSM.

6. Conclusions and Remarks

This chapter presented the self-organizing approach to solve the surface reconstruction problem. The features of the Self-Organizing Map and of some of its well known variants, such as the Topology Representing Networks, the Growing Cell Structures and the Growing Neural Gas, were presented. Also, two novel SOM based models were presented – GSRM and GSOSM - which have been proposed by the authors and overcome several drawbacks of their predecessors.

GSRM is a self-organizing based surface reconstruction method that profits from the topology learning ability and incremental growth of Growing Neural Gas, while, at the same time, it also modifies some of the standard GNG operations to meet some requirements of the surface reconstruction method, for example: to produce 2-manifold meshes of triangles and to have faces that are approximately equilateral. The main advantages of GSRM are: it can learn the topology of a given object from the input point cloud; it produces meshes with different resolutions during the learning process, avoiding the need for a mesh simplification step. The main limitations of GSRM are: a dense, random sample set of points is required for the reconstruction.

The GSOSM explores the concept of Voronoi region to define a new learning rule (CCHL) that is employed as the basis for establishing connections between nodes. The CCHL enables GSOSM to produce a complete local triangulation, thus overcoming the limitation of its predecessor (CHL). The two step adaptation operator of GSOSM qualifies it to handle any input sequence producing a mesh that is faithful to the objective surface. Moreover, the GSOSM algorithm overcomes the homeomorphism limitations and is able to produce representations of folded surfaces that may or may not be 2-manifold. However, the GSOSM is not able to produce a quality mesh if the input has areas with low sampling density.

Moreover, there was discussion of pre and post-processing steps and some of the other procedures that need to be observed when using the models cited as a tool for solving the surface reconstruction problem.

The methods presented here as solutions for the surface reconstruction problem present promising results. However, they still have some limitations such as to deal with sparse or not randomly sampled point clouds. Therefore this is an open area for future developments.

7. References.

Amenta, N., Choi, S. & Kolluri, R. K. (2001) The power crust, *In Proceedings of the sixth ACM symposium on Solid modeling and applications*, pp. 249-266.

Aurenhammer, F. & Klein, R. (2000) *Voronoi diagrams*, In J.-R. Sack, J. Urrutia (Eds.), Handbook of Computational Geometry, Elsevier Science/North-Holland, Amsterdam, pp. 201-290.

Barhak, J. (2002) *Freeform objects with arbitrary topology from multirange images,* PhD thesis, Technion - Israel Institute of Technology, Haifa, Israel.

de Berg, M., Cheong, O., van Kreveld, M. & Overmars, M. (2008) *Computational Geometry: Algorithms and Application,* 3rd Edition, Springer-Verlag.

Bernardini, F., Bajaj, C. L., Chen, J. & Schikore, D. (1999) Automatic Reconstruction of 3D CAD Models from Digital Scans, *International Journal of Computational Geometry and Applications*, Vol 9, No 4, pp. 327-369.

Bernardini, F., Martin, I., Mittleman, J., Rushmeier, H. & Taubin, G. (2002) Building a digital model of Michelangelo's Florentine Pieta, *IEEE Computer Graphics and Applications*, Vol 22, No 1, pp. 59–67.

Bobenko, A. I. & Springborn, B. (2007) A discrete Laplace-Beltrami operator for simplicial surfaces, *Discrete and Computational Geometry*, Vol 38, No 4, pp. 740-756.

Brito, A. D., Doria, A. D., de Melo, J. D. & Goncalves, L. M. G. (2008) An adaptive learning approach for 3D surface reconstruction from point clouds, *IEEE Transactions on Neural Networks*, Vol 19, No 6, pp. 1130-1140.

Cignoni, P., Rochini, P. & Scopigno, R. (1998) Metro: Measuring Error on Simplified Surfaces, *Computer Graphics Forum*, Blackwell Publishers, Vol 17, No 2, pp. 167-174.

DalleMole, V.L. & Araujo, A. F. R. (2008a) The growing self-organizing surface map. *In International Joint Conference on Neural Networks*, pp. 2061-2068.

DalleMole, V. L. & Araújo, A. F. R. (2008b) The Growing Self-Organizing Surface Map: Improvments, *In 10th Brazilian Symposium on Artificial Neural Networks (SBRN´08)*, Salvador.

DalleMole, V. L. & Araujo, A. F. R. (2009) Growing Self-Organizing Surface Map: learning a surface topology from a point cloud. To appear on Neural Computation.

Do Rêgo, R. L. M. E., Araujo, A.F.R. & de Lima Neto, F.B. (2007) Growing self-organizing maps for surface reconstruction from unstructured point clouds, *In International Joint Conference on Neural Networks*, pp. 1900–1905.

Do Rêgo, R. L. M. E., Bassani, H.F. & Filgueiras, D. (2009) Surface Reconstruction System Based on a Growing Self-organizing Map. *To appear in International Conference on Artificial Neural Networks*.

Edelsbrunner, H., Kirkpatrick, D.G. & Seidel, R. (1983) On the shape of a set of points in the plane, *IEEE Transactions on Information Theory*, Vol 29, pp. 551-559.

Fisher, M., Springborn, B., Bobenko, A. I. & Schroder, P. (2006) An algorithm for the construction of intrinsic Delaunay triangulations with applications to digital geometry processing. *In SIGGRAPH '06: ACM SIGGRAPH 2006 Courses*, pp. 69-74, New York, NY, USA, ACM.

Fritzke, B. (1996) *Unsupervised ontogenetic networks*. Handbook of Neural Computation.

Fritzke, B. (1995) A Growing Neural Gas Network Learns Topologies, *in Advances in Neural Information Processing Systems* 7, G.Tesauro, D.S. Touretsky and T. K. Leen, MIT Press, Cambridge MA.

Fritzke, B. (1994) Growing cell structures - A self-organizing network for unsupervised and supervised learning, *Neural Networks*, Vol 7, pp. 1441-1460.

Hoffmann, M. & Varady, L. (1998) Free-form modelling surfaces for scattered data by neural networks, *Journal for Geometry and Graphics*, Vol 2, No 1, pp. 16.

Hoppe, H., Derose, T., Duchamp, T., Mcdonald, J. & Stuetzle W. (1992) Surface reconstruction from unorganized points, *In Proceedings of Siggraph Conference*.

Ivrissimtzis, I., Jeong, W. K., Lee, S., Lee, Y. & Seidel, H. P. (2004) Neural Meshes: Surface reconstruction with a learning algorithm. *Eurographics Symposium on Point-Based Graphics*, pp. 1-10.

Ivrissimtzis, I., Jeong, W. K., Lee, S., Lee, Y. & Seidel, H. P. (2004) Surface Reconstruction Based on Neural Meshes, *Matematical methods for CAGD*, Nashoboro Press, Brentwood.

Jockusch, J. & Ritter, H (1993) An instantaneous topological mapping model for correlated stimuli. *In International Joint Conference on Neural Networks*, Vol 1, pp. 529-534.

Kohonen, T. (1998) The self-organizing map, *Neurocomputing*, Vol 21, pp. 1-6.

Mäntylä, M. (1988) *An Introduction to Solid Modeling*. Computer Science Press, Rockville.

Martinez, T., Schulten, K. (1994) Topology Representing Network, *Neural Networks*, Vol 7, No 3, pp. 507-522.

Martinez, T. (1993) Competitive Hebbian learning rule forms perfectly topology preserving maps. International Conference Artificial Neural Networks, pp. 427-434.

Martinez, T., Schulten, K. (1991) A "Neural-Gas" Network Learns Topologies, *in Artificial Neural Networks*, T. Kohonen, K. Makisara, O. Simula and J. Kangas, Elselvier Science Publishers. North-Holland.

Miller, J.V., Breen, D.E., Lorensent, W.E., O'Bara, R.M. & Wozny, M.J. (1991) Geometrically deformed models: A method for extracting closed geometric models from volume data. *In Proceedings of the International Conference on Computer Graphics and Interactive Techniques*, Vol 25, pp. 217-226.

Qin, H., Mandal, C. & Vemuri, B.C. (1998) Dynamic Catmull-Clark subdivision surfaces, *IEEE Transactions on Visualization and Computer Graphics*, Vol 4, No 3, pp. 215-229.

Saleem, W. (2003) *A Flexible framework for learning-based surface reconstruction,* Master's thesis, Computer Science Department, University of Saarland, Saabrucken, Germany.

Satava, R. M. & Jones, S. B. (1998) Current and future applications of virtual reality for medicine". Proceedings of the IEEE, Vol 86, No 3, pp. 484-489.

Yu, Y. (1999) Surface reconstruction from unorganized points using self-organizing neural networks, *In Proceedings of IEEE Visualization Conference*, pp. 61-64.

12

Self-Organizing maps for processing of data with missing values and outliers: application to remote sensing images

Bassam Abdel Latif

Abstract
This chapter presents how to recover a data set that contains missing values, errors and outlier values using the Self-Organizing Maps (SOM). It has been shown by many authors that if a data set contains missing values (missing components of some observations), then the SOM is a good candidate to recover it. The idea is as simple as to use the center of each subclass to estimate the missing values of a given observation. The virtue of the SOM regarding this problem is two folded: firstly, it is a non-parametric regression procedure that does not suppose any underlying models of the data set, and secondly it uses the information from similar observations to refine the positions of subclasses centers and hence gives better estimation.
Therefore, the SOM for missing value will be detailed first, and the modification of this algorithm will be proposed through the introduction of a new similarity measure (replacing the Euclidean distance, the widely used one with SOM applications) that is to be used to match different observations to their best matching neurons.
These algorithms will be presented by the help of a case study that recovers the missing and erroneous values in a set of remote sensing images (due to the presence of clouds and shadows). The application of the SOM algorithm to recover the missing data in a heavily cloudy region in France, monitor its superiority to other methods reported in the literature.

1. SOM for incomplete data

As we have seen in previous chapters, the SOM is used to cluster a set of observations $\mathbf{X} = \mathbf{x} : \mathbf{x} = x_1, \ldots, x_k, \ldots, x_n$ into a set of M clusters or classes of $\mathbb{R}^n$. The clustering is as good as the separability found between clusters. Normally, each cluster is represented by more than one neuron in the SOM output space to account for the natural variability in each cluster. Therefore, the number of neurons in the SOM, M, is usually greater than the natural number of clusters or classes found in the data set $\mathbf{X}$. Hence neighboring neurons share a considerable amount of information between them. This virtue of the SOM is used to overcome the reality that some observations, x, may have incomplete set of components, x_k, due to different reasons: insufficient data, sensor failure, experiments non-completed, etc.
Authors of Fessant & Midenet (2002) showed how the SOM algorithm may be used to estimate, recover, missing values in surveys. In Cottrell & Letrmy (2005), authors used the same method to estimate missing values in a socio-economical database. The basic ideas in their

work are: firstly, using valid components only in the training of the SOM and secondly using the synaptic weights of each neuron to estimate the missing components of the corresponding input observation. The following paragraphs will describe this process in more details.
Due to the huge size of the data used in many applications, we usually use two data sets in the clustering (or missing values estimation) using the SOM algorithm: the original one, $\mathbf{X}$, to be clustered and a representative subset of it, $\mathbf{X}'$. This last subset is smaller in size than the original set, this accelerates the training process. In the case that the original set, $\mathbf{X}$, contains observations with missing values, the training subset, $\mathbf{X}'$, may or may not contain observations with missing values. To proceed in the training phase, if an input observation x contains missing component, x_k, the set M_x is introduced to define the indices of these missing values. In this case, M_x is a sub-set of $1,2,\ldots,n$.
In the training phase, the winning neuron at iteration t, $\mathbf{C}_{m_x}(t)$, is selected in responding to the input $\mathbf{x}$ using the equation:

$$\|\mathbf{x} - \mathbf{C}_{m_x}\| = \min_{m\in\{1,\ldots,M\}} \|\mathbf{x} - \mathbf{C}_m\|. \tag{1}$$

When facing incomplete vector, $\mathbf{x}$, the Euclidean distance $\|\mathbf{x} - \mathbf{C}_m(t)\|$ is computed with the valid components of $\mathbf{x}$ only ($x_k \notin M_x$). The weight updating of neurons (the best-matching neuron, $\mathbf{C}_{m_x}$, and its neighbors that belong to the set $N_{m_x}(t)$) affects the valid components only. In other words, by denoting $\mathbf{C}_m(t) = (c_{m;1},\ldots,c_{m;k},\ldots,c_{m;n})^t$ the components of weight vector associated to the neuron $\mathbf{C}_m$ at instant t and $x = (x_1,\ldots,x_n)^t$, then the weight updating process is accomplished by the equation:

$$c_{m;k}(t+1) = c_{m;k}(t) + h_{m,m_x}(t)\Big[x_k - c_{m;k}(t)\Big], \tag{2}$$

for $k \notin M_x$ (*i.e.* for valid components). Otherwise, no modification is performed:

$$c_{m;k}(t+1) = c_{m;k}(t). \tag{3}$$

1.1 Estimation of missing values

One of the interesting properties of the SOM algorithm for missing values is that it allows an *a posteriori* estimation of these missing values. Once the SOM has been trained, the missing values may simply be estimated by using:

$$\hat{x}_k = c_{m_x;k} \quad k \in \mathbf{M}_x. \tag{4}$$

When the Kohonen algorithm converges with a neighborhood of length 0, it is known that the code-vectors $\mathbf{C}_m$ converges asymptotically to the mean value of its class or subclass m. Therefore, this estimation method consists in estimating the missing values of a random variable by the mean value of its class, defined through a training set. It is obvious that the more the compactness and the separability of the classes, the more accurate the estimation. Eq. (4) may be turned to a fuzzyfication by using membership values of the observation x to the set of the code vectors Cottrell & Letrmy (2005). These membership values may also provide confidence intervals Cottrell & Letrmy (2005).

2. Case study

In this section we will study the application of this technique to a set of MODIS (Moderate Resolution Imaging Spectroradiometer) data dedicated to identifying bare soils in the Brittany region of France during the winter season. Most of observations are unusable due to the presence of clouds or shadows (knowing that this region is characterized by 200 rainy days per year). Therefore, in order to make bare soil monitoring in such conditions, it is necessary to process as most data as possible.
In this application, MODIS data are collected in time series of reflectances (near-infrared and red reflectances). In other words, each pixel in a multiband file is a temporal profile of a certain reflectance channel. The temporal profile of each pixel is considered to as an observation that contains as much components, x_k, as the number of dates, n, in the time series. Therefore, the SOM algorithm for the missing values will be applied twice on two different sets: one set for the near infrared channel of the MODIS image series, $\mathbf{X}_{NIR}$, that contains observations of the form $\mathbf{x} = NIR_1, \ldots, NIR_k, \ldots, NIR_n$, and another set for the red channel, $\mathbf{X}_R$, that contains observations of the form $\mathbf{x} = R_1, \ldots, R_k, \ldots, R_n$
If one temporal profile has cloud or shadow contamination in any date, then the index of this date is assigned to the set M_x and the whole observation is marked as having erroneous values. These erroneous values have to be detected and marked as missing values before proceeding with the SOM algorithm for missing values to recover them.

2.1 Data

We describe in this section the data used in this case study. These data contains two types of satellite data. The first type is the MODIS data that are contaminated by the presence of clouds and their associated shadows, while the second type is a set of high resolution data that has been used in the validation of the SOM algorithm for missing values. The following paragraphs provide more details about these data.

2.1.1 MODIS data

MODIS images of the season 2002-2003 were used in this case study. They were selected according to sensor zenithal viewing angle and cloud coverage criteria. Despite available cloud cover information in each MODIS tile, images have been selected visually. Selection based on MODIS tiles information appears to be too imprecise, since the information is not spatially distributed. Images with less than 50% of cloud coverage are selected. Concerning zenithal viewing angle selection, images acquired with an orbit track centered on Brittany and a radius of 200 km were selected. Low zenithal viewing angle values, inferior to 20, were selected only to avoid spatial resolution variations.
10 images are available from 11-25-2002 to 4-16-2003. Almost all of the images captured in these dates are contaminated by clouds, even the fact that these dates were selected, indeed, for their minimal amount of cloud coverage. Only band 1 and 2 of the MODIS data have been processed due to their 250m spatial resolution. Therefore, the SOM algorithm for missing values will be applied twice: one process dedicated to the data set constituted by the 10 dates of red reflectances and another independent process applied to the data set constituted by the 10 dates of near-infrared reflectances. In this study, processing steps dedicated to recover the erroneous data in the near-infrared channel is presented. The method can be applied in the same way to any other reflectance channel. Fig. 1 shows the near-infrared band at 11-25-2002 over the Brittany region.

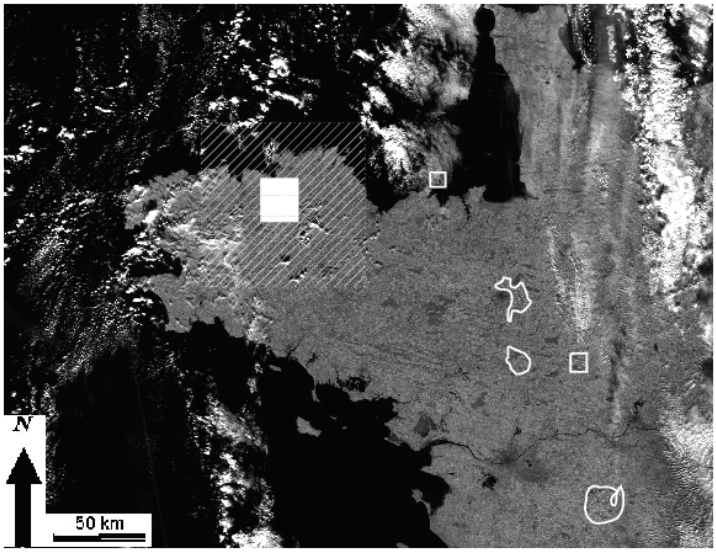


Fig. 1. Typical examples of data affected by clouds: the Brittany region as captured by MODIS near infrared band. Polygons represent areas that have been selected for training (these areas are almost clouds free for the temporal series used in the study). Rectangle hashed area shows the location of SPOT/HRVIR image acquired on 01-24-2003. White area corresponds to the deleted data of 01-24-2003 (see sec. 3.1).

In order to assess results of the recovering process, some original MODIS data has been replaced by missing values in the image of 01-24-2003. This date has been chosen because there is no clouds or shadows in the selected area and that high resolution data corresponding to the same area is available. The validation area is about 25×20km (see the white rectangle in Fig. 1). This validation area has been converted from reflectance values to zero values that are close to the deep shadow values.

2.1.2 High resolution data

Two high resolution images are used for validation. SPOT/HRVIR and Landsat/ETM+ images were corrected from atmospheric effects by using the 5S model Tanre et al. (1990) and then corrected from geometric distortion. High resolution images were aggregated to 250m resolution, by using the Point Spread Function of the MODIS sensor Huang et al. (2002), for the comparison to the original MODIS data. The accuracy of the recovering of missing data could have been evaluated by using original MODIS data. Nevertheless, data from alternate sensors is used indeed to evaluate the sensitivity of the recovering process with regards to the atmospheric and geometric difference introduced by those alternate sensors.

2.2 The use of the SOM

2.2.1 Finding outliers

We suppose here that within a temporal profile of normal cover, an outlier value represents an erroneous value (presence of clouds or shadows). This hypothesis may not be acceptable for other types of data, e.g. in high resolution data where a white car or a bright roof may induce

an outlier-like pixel. Therefore, a simple algorithm, to identify outliers in each temporal profile is used in this study. For operational use, however, one can use an appropriate cloud detection algorithm for the processed data, e.g. the standard MODIS 1km cloud mask.
One can use any classical test (for which an interest comparison may be found in Bakar et al. (2006)) to carry out this task. Nevertheless, most of these tests are based on the normal distribution assumption, which is no longer the case here. In fact, clouds and shadows may be thought of as *salt and pepper* noise. Then, the median filter should be more appropriate for this kind of noise. Therefore, the Box and Whisker method has been used, since it does not depend on a statistical model.
The technique states that x_k is mostly an outlier at date k if:

$$|x_k - 1.5\,(\mathbf{x}_{3/4} - \mathbf{x}_{1/4})| > |\mathbf{x}_{1/2}|. \tag{5}$$

It is based on rank statistics where: $\mathbf{x}_{1/2}$ is the median and $\mathbf{x}_{1/4}$ (resp. $\mathbf{x}_{3/4}$) is the first (resp. third) quartile of the temporal signature x. One has to note that outliers are detected (and then removed) at a given date only. Whenever an outlier is detected at one date or at several dates, the temporal signature is preserved and associated to the set of non-empty missing components $\mathbf{M}_x$.
This simple method could be improved, but satisfying results have been obtained when applied to the MODIS data set because it is capable of detecting both clouds and shadows and, hence, overcomes the standard MODIS 1Km cloud mask which does not detect shadows.

2.2.2 SOM implementation

A 50×20 neuron map has been implemented and trained on a specific area (shown in figure 1) where few clouds have been found in the time series.
The number of neurons in the SOM grid is a compromise between the processing time and the required quantization error (the average distance between each vector in the test data and its Best Matching Unit (BMU) in the SOM grid). One can begin by a reasonably small map, say 12×8, and increases the size of the map to reach a satisfactory quantization error. In this application, we reached an average quantization error of 0.0327 (in the sense of Mean Square error, MSE) with the proposed map size in a reasonable time: 45 sec. It is also advisable to have a rectangular shape map if the data has a correlation between its different components. So the choice of the rectangular shape 50×20 of the SOM grid.

2.2.3 Data projection on the SOM grid

Once the SOM grid has been trained until convergence, the complete time series has to be processed. Each pixel in the original data is projected on the SOM grid to find the weight vector that best matches it. The word projection is used here to indicate that the output image contains a subset of reflectance values that have been yielded by the SOM algorithm. It is worth mentioning here that the 50×20 neurons of the SOM grid are representative enough for each class, *i.e.* the variability of inter-class was taken into account.
Two different similarity measures have been used to find the winner neuron, that matches a given observation, in the projection phase. The Euclidean distance and the *spectral*[1] angle between each vector C_m in the SOM grid and the input vector x. The Euclidean distance was found to be more appropriated distance measure to recover the contaminated data by projection on the SOM grid. Further discussion about similarity measures will be follow in section 4

[1] It is a *temporal* application of the *Spectral* angle.

It is interesting to stress that the time series does not have to be uniformly sampled over the time. The SOM algorithm is dealing with vectors of $\mathbb{R}^{10}$ with no consideration to the gaps between those 10 dates.

2.3 Validation

Two methods have been used to validate recovered reflectance data. The first one is a simple difference between the so-called recovered image and the original one. The second method consists in mapping the residual errors of the linear regression between the recovered data and the original one. The linear relation is of the form:

$$y = ax + b \tag{6}$$

where x takes the value of the original MODIS pixels and y is associated to the pixels of the recovered image. In order to evaluate difference between the original image and the recovered one, residual errors (as denoted to as *residual* in the text) are calculated and mapped. Residuals express the distance between values of the recovered image and values estimated with the linear model:

$$\epsilon_i = y_i - ax_i - b \tag{7}$$

where i corresponds to a pixel index.
This method was originally used to detect changes with unscaled values Ingram et al. (1981); Jha & Unni (1994). In this study, this method is used to compare recovered MODIS image with high resolution images.

3. Results

Results yielded by the used method are exposed in two steps:

- validation of reflectance values after processing;
- validation of recovered spatial structures.

A comparison with the compositing method that deals with the problem of clouds and shadows in the low resolution images and which is the most used method in the relevant literature, is conducted in Abdel Latif et al. (2008).

3.1 Validation of reflectance values after processing

In this part, results are compared by considering:

- the projection of contaminated observations directly onto the SOM grid, referred to as SOM1 algorithm,
- the use the Box and Whisker technique to isolate outliers (erroneous values in each observation) and mark them as missing values before projection onto the SOM grid, referred to as SOM2 algorithm (cf. Fig. 2)

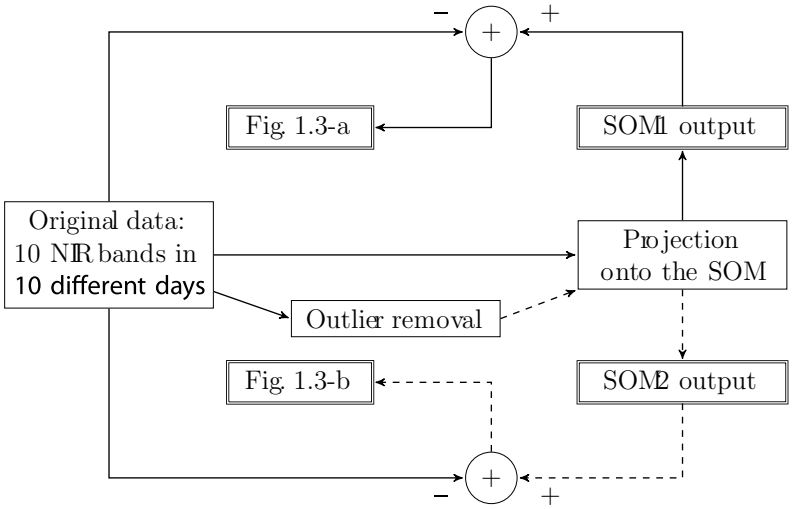
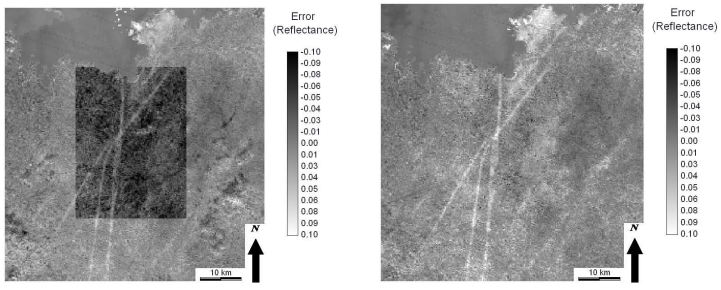


Fig. 2. A schematic representation for SOM1 and SOM2 algorithms and different processing steps that have been used to produce Fig. 3.

A simple difference is calculated to confirm the ability of SOM algorithms to recover reflectance values, (cf. Fig. 2 and 3). The visual analysis of Fig. 3 shows clearly that recovered images become more homogeneous when using SOM2 algorithm. Fig. 3-(a) shows that using SOM1 algorithm, recovered reflectance values of the artificial no-data rectangle (the white rectangle in Fig. 1) are lower than the rest of the image. Fig. 3-(b) shows the recovered reflectance values: no discontinuities may be distinguished between the no-data rectangle and its surroundings when SOM2 algorithm is used.



(a) (b)

Fig. 3. Difference between recovered MODIS image and original MODIS image, 01-24-2003. (a) with SOM1 algorithm; (b) with SOM2 algorithm.

Considering that the difference between original reflectance values and recovered ones are associated to errors, Fig. 4 shows the distribution of these errors obtained when processing the area defined on Fig. 1. It is worth noting that, with SOM1 algorithm, reflectance values are underestimated with a mean around 0.05 (the mean of error distribution is at about -0.05).

When using SOM2 algorithm, the mean of errors is around 0.02. Moreover, 80% of errors are located between -0.04 and 0.07.

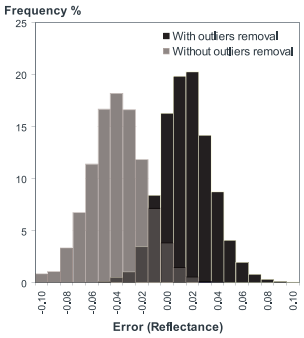
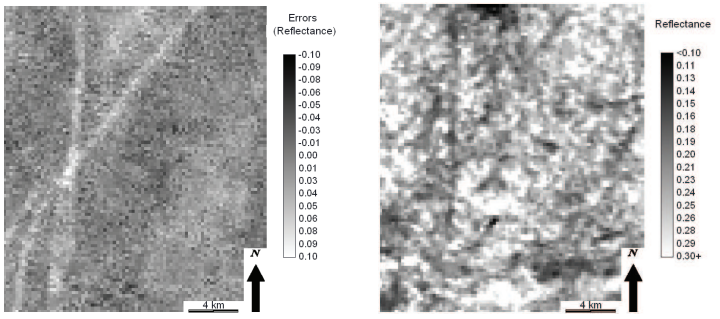


Fig. 4. Distribution of errors between recovered MODIS image and original MODIS image, evaluated on the no-data rectangle, on 01-24-2003.

Fig. 5-(a) focuses on some atmospheric artifacts seen in Fig. 3: white vertical and diagonal lines. These lines are whiter in the difference image, Fig. 5-(a), which means that the SOM algorithm chose whiter samples (weight vectors) to replace darker ones in the original MODIS image Fig. 5-(b). These lines are due to shadows of airplane trails which were not visually detectable from the NIR band but from the visible bands only.



(a) (b)

Fig. 5. Atmospheric artifacts (shadows of airplane trails) corrected by the SOM algorithm, 01-24-2003. (a) Difference between recovered MODIS image using SOM2 and original MODIS image; (b) Original MODIS image.

These results show that, by using the Box and Whisker technique to isolate contaminated values, the retrieved reflectance values are scaled and located in the range of valid reflectance values. In the next part, the study focuses on the spatial structure of recovered data.

3.2 Validation of recovered spatial structures

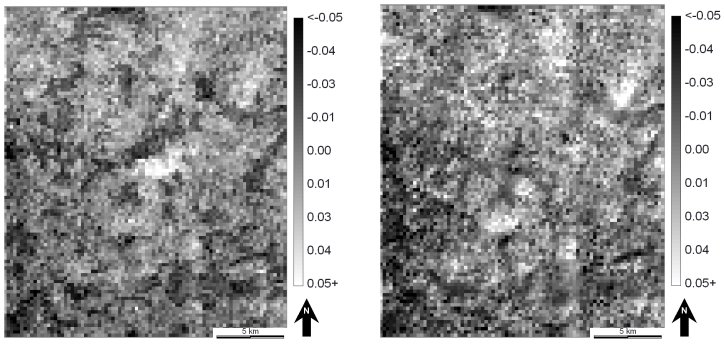
The next stage of the validation points up the correctness of recovered spatial structures by comparing the recovered images with 1) original MODIS images and 2) high resolution images aggregated at 250m spatial resolution.

The correlation coefficients between the original MODIS image, SPOT/HRVIR image and the recovered MODIS image, all on 01-24-2003, are shown in table 1. It appears that isolating contaminated values, by the Box and Whisker method, before the projection on the SOM grid increases the correlation between recovered MODIS and original MODIS (from 0.83 to 0.87) and increases the correlation between recovered MODIS and SPOT/HRVIR (from 0.81 to 0.86). Residuals of the two linear relations highlight the atmospheric artifacts that have been substituted by the SOM algorithm. Fig. 6 shows the residuals of the relation between the recovered MODIS image and the SPOT/HRVIR image. Whiter colors mean that these values were recovered to higher values than the simple linear relation, of eq. (6), did (cf. section 2.3). Darker colors mean that SOM algorithm estimated values lower than the simple linear relation did. Hence, these differences in the estimated values are mainly due to difference of atmospheric contamination such as haze and shadows, and show the relevance of the SOM estimation.

Fig. 7 shows the residuals of the relation between original and recovered MODIS image. As before, whiter colors means that these pixels were estimated by the SOM algorithm to higher values than in the original image.

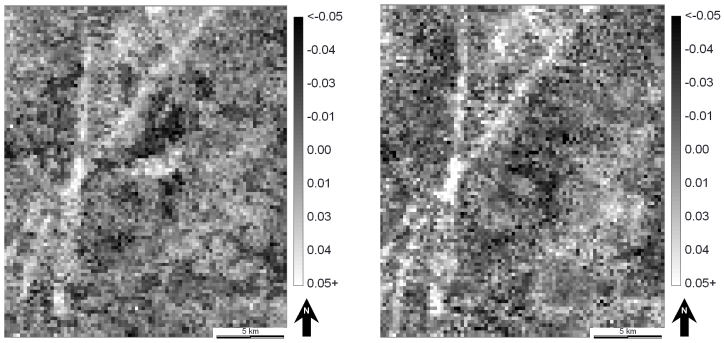
r	original MODIS	recovered MODIS using algorithm:	
		SOM1	SOM2
SPOT/HRVIR	0.84	0.81	0.86
original MODIS	1	0.83	0.87

Table 1. Correlation coef. between recovered MODIS data and SPOT/HRVIR data, 01-24-2003.



(a) (b)

Fig. 6. Maps of residuals between recovered MODIS image and SPOT/HRVIR image, 01-24-2003. (a) SOM1 algorithm; (b) SOM2 algorithm.



(a) (b)

Fig. 7. Maps of residuals between recovered MODIS image and original MODIS image, 01-24-2003. (a) SOM1 algorithm; (b) SOM2 algorithm.

To confirm these results, recovered image was compared to Landsat/ETM+ and MODIS images with a larger area to evaluate how will be the correlation coefficient for spatially large areas. According to table 2 the relation between the original MODIS and the Landsat/ETM+ image shows a correlation coefficient of 0.91. The recovering process decreases slightly the relation with the aggregated Landsat/ETM+ image with the SOM2 algorithm (from 0.91 to 0.89) and with the SOM1 algorithm (from 0.91 to 0.88). This decrease in the correlation with the Landsat/ETM+ image is interpreted by the fact that this image is not as clear as the SPOT/HRVIR image. Therefore, the recovery process changes some values in the MODIS image which were well correlated, as clouds or shadows, in the Landsat/ETM+ image. Residuals between recovered MODIS image and Landsat/ETM+ image are shown in Fig. 8.

r	original MODIS	recovered MODIS using algorithm:	
		SOM1	SOM2
Landsat/ETM+	0.91	0.88	0.89
original MODIS	1	0.96	0.97

Table 2. Correlation coef. between recovered MODIS and Landsat/ETM+ data, 03-15-2003.

The comparison of results, obtained from 01-24-2003 and 03-15-2003 cases, proves that the use of an outlier detector and removal, SOM2 algorithm, improves the performances of the non-parametric estimator.

Using residuals of a linear relation to validate recovered reflectance values, the spatial structure of recovered images is considered to be corresponding to the landscape structure. These analysis allow to conclude that the spatial structure is well recovered. Moreover, the SOM algorithm removes small atmospheric artifacts that are not visually detectable and replaces them by valid reflectance values (cf. Fig. 5).

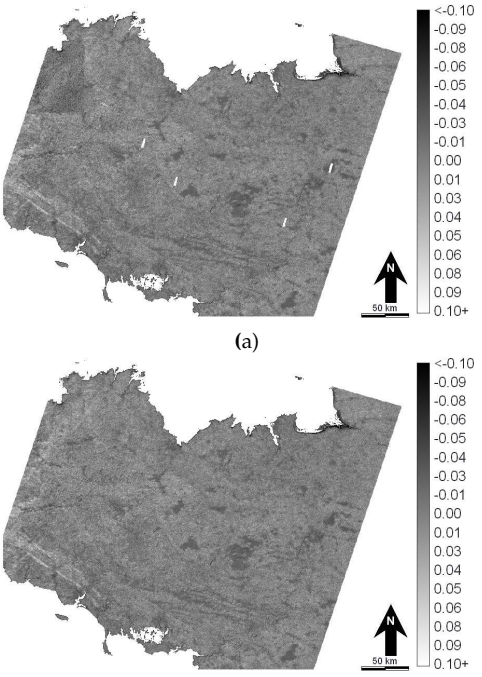


Fig. 8. Maps of residuals between recovered MODIS image and Landsat/ETM+ image, 03-15-2003. (a) SOM1 algorithm; (b) SOM2 algorithm.

4. An appropriated similarity measure

In section 3, we saw that results obtained using the SOM2 algorithm is better than those obtained by the SOM1 algorithm. This is because in the projection phase, SOM2 algorithm uses only the valid components in searching the best matching temporal profile, from the SOM grid, for a given contaminated temporal profile. Unfortunately, the performance of the SOM2 algorithm will be identical to that of SOM1 algorithm if the outlier detector fails to detect the outlier values in the temporal profile. The question now is: are there any better similarity measures that better match the contaminated profiles with the code vectors? In other words, is there any similarity measure that is robust against the presence of outliers in the temporal profile, before the projection onto the SOM, even with the use of the simple outlier detector of Box and Whisker?

We present hereafter a similarity measure that will make the SOM2 algorithm robust against the presence of any erroneous values that remain without detection prior to the projection phase. This robustness makes it possible to apply the SOM algorithm for missing values without the need to the intermediate stage of detecting and marking erroneous values to recover these erroneous values.

4.1 Similarity measures

We can find 4 different similarity measures that are heavily used in the literature of remote sensing applications (the thematic field of the case study). They are: (a) the Euclidean distance, (b) the spectral angle mapper, (c) the spectral correlation measure and (d) the spectral information divergence measure and (e) a proposed similarity measure.

4.1.1 Euclidean distance

The Euclidean distance (ED) between two temporal profiles portrayed as vectors $(\mathbf{x})$, $(\mathbf{y})$ lie in $(\mathbb{R}^n)$ is given by:

$$\mathrm{ED}(\mathbf{x},\mathbf{y}) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}. \tag{8}$$

As a similarity measure, we can drop out the square root function because it is a monotonic increasing function. Its presence or absence will not affect the results.

4.1.2 Spectral angle measure

The Spectral Angle Measure (SAM) Kruse et al. (1993), will be applied to a temporal profiles and not to a spectral signatures, is the measure of the angle between two vectors $(\boldsymbol{x})$ and $(\boldsymbol{y})$ which lie in $(\mathbb{R}^n)$ and is as follows:

$$\mathrm{SAM}(\boldsymbol{x},\boldsymbol{y}) = \cos^{-1}\left(\frac{\sum_{i=1}^{n} x_i y_i}{[\sum_{i=1}^{n} x_i^2]^{1/2}\sum_{i=1}^{n} y_i^2]^{1/2}}\right). \tag{9}$$

The difference between the Euclidean distance and spectral angle is that the later is not affected by the magnitude of the involved vectors (if two vectors have the same direction, then it will not matter if their magnitudes are different).

4.1.3 Spectral correlation measure

The Spectral Correlation Measure (SCM) van der Meer (2006) is calculated as the correlation coefficient between two temporal profiles ($\mathbf{x}\mathrm{and}\mathbf{y} \in \mathbb{R}^n$) as:

$$\mathrm{SCM}(\mathbf{x},\mathbf{y}) = \frac{n\sum_1^n x_i y_i - \sum_1^n x_i \sum_1^n y_i}{\sqrt{[n\sum_1^n x_i^2 - (\sum_1^n x_i)^2][n\sum_1^n y_i^2 - (\sum_1^n y_i)^2]}}. \tag{10}$$

This similarity measure has the advantage that it takes into account the relative shape of the two vectors as well as the component matching. The correlation can be both positive and negative. Such a measure takes brightness difference and shape difference between vectors into account.

4.1.4 Spectral information divergence

The Spectral Information Divergence (SID) measure Chang (2000) calculates the distance between the probability distribution produced by two vectors $(\mathbf{x})$ and $(\mathbf{y})$ (here, temporal profiles) by firstly found the probability distribution $(\mathbf{p} = \{p_i\}_1^n,\ p_i = x_i/\sum_1^n x_i)$ and $(\mathbf{q} = \{q_i\}_1^n,\ q_i = y_i/\sum_i^n y_i)$. Then the spectral information divergence is given by:

$$\mathrm{SID}(\mathbf{x},\mathbf{y}) = D(\mathbf{x}\|\mathbf{y}) + D(\mathbf{y}\|\mathbf{x}). \tag{11}$$

Where $(D(\mathbf{x}\|\mathbf{y}) = \sum_1^n p_i \log(p_i/q_i))$ and $(D(\mathbf{y}\|\mathbf{x}) = \sum_i^n q_i \log(q_i/p_i))$. It should be noted that $(D(\mathbf{x}\|\mathbf{y}))$ is called the relative entropy of $(\mathbf{y})$ with respect to $(\mathbf{x})$ which is also known as Kullback-Leibler information function.

4.1.5 A proposed similarity measure

This subsection presents a similarity measure for the first time in the literature to be used with the SOM algorithm, that is why we use the word "proposed" in this book chapter. The proposed similarity measure is suitable when searching a candidate codebook for a given vector that has some erroneous components. This measure tries to decrease the effects of erroneous components to the resulted measure with respect to the non-erroneous components. The proposed similarity measure was firstly used in Chapelle et al. (1999) as a heavy-tailed radial base function (RBF) that has been used in a classification of data base images by comparing their histograms using the Support Vector Machine (SVM) technique. This non-Gaussian RBF kernel is given by:

$$K(\mathbf{x},\mathbf{y}) = e^{-\rho \sum_i |x_i^a - y_i^a|^b}, \tag{12}$$

where $(a \leq 1)$ and $(b \leq 2)$. Since the exponential function is a monotonic function, hence its presence or absence will not affect the comparison between a vector $(\mathbf{x})$ and a set of vectors $(\mathbf{y}_i)$ to choose the best match vector $(\mathbf{y}^*)$ to the given input vector $(\mathbf{x})$. Therefore, the proposed similarity measure is given by:

$$\mathrm{D}(\mathbf{x},\mathbf{y}) = \sum_{i=1}^{n} |x_i^a - y_i^a|^b. \tag{13}$$

It remains the determination of the two parameters a and b. For the special case of $a = 1$ and $b = 2$ this similarity measure performs exactly as the Euclidean distance similarity measure.

4.2 The proposed similarity measure

We will proceed to determine best values for the parameters a and b in an empirical way. This empirical way uses a set S_{MODIS} of 5000 MODIS temporal profiles. Each temporal profile consists of 10 dates of near-infrared reflectance values. These temporal profiles have been selected randomly from the reconstructed MODIS time series that has been yielded in section 2. To come over the quantization effects of the SOM algorithm for missing values, a random reflectance value between -0.02 and 0.02 has been added to all reflectance values in the S_{MODIS}. Reflectance values in this set are supposed to match real objects on the land surface.

A perturbed set , S_p, has been simulated using the S_{MODIS} set. From the 5000 temporal profiles in each band in S_{MODIS}, 1500 locations (at each individual date) are selected randomly to simulate perturbed reflectance values and have been assigned to a random value between 0 and 0.9. The 1500 locations in each date are independent of other 1500 locations in other dates. In this way, the set S_p has 4733 temporal profiles that have perturbed values in at least one date. There are only 20 temporal profiles that has a perturbation in more than 6 dates with 17 one perturbed at 7 positions, 3 at 8 positions and 1 temporal profile that has a perturbation at 9 dates.

Therefore, the best values for a and b parameters will be those which maximize the correct matching from the set S_p to the set S_{MODIS} (if a perturbed temporal profile is mapped to its original, non perturbed temporal profile, this mapping or matching is considered as to be correct). Fig. 9 shows the percentage of correct matching in function of one parameter, here variable, of a or b while fixing the other parameter to 1. The figure shows also that at a and b equal 1, the percentage of correct matching is 23.86%. For $a = 1$ and $b = 2$ (Euclidean distance similarity measure) the percentage of correct matching is only 13.04%.

If the value of a or b get larger than 1, the performance is dramatically decreased. For values less than 1, the percentage of correct matching increases. At $a = 0.1, b = 1$ the correct matching

is 48.98%, while at $a = 1$ and $b = 0.1$ the correct matching is 99.92%. In other words, the similarity measure with $a = 1$ and $b = 0.1$ has correctly matched 4996 temporal profiles from the 5000 one in the test. That is to say this arrangement has succeeded to correctly match temporal profiles perturbed at up to 70% of its components. The rest 4 temporal profiles that have not been matched correctly are the 3 temporal profiles that have been perturbed in 8 out of 10 dates (80% of the data are wrong) and the temporal profile that have been perturbed in 9 out of 10 dates.

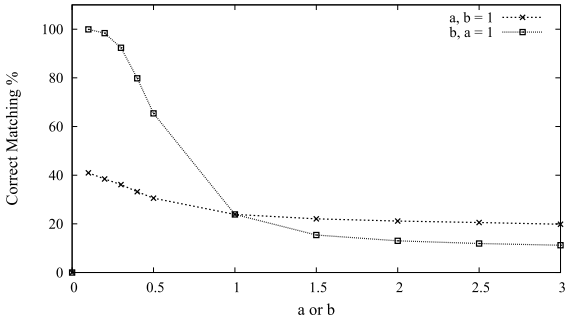


Fig. 9. The percentage of correct matching as a function of one of the parameters a or b while fixing the other at 1.

We notice also from Fig. 9 that decreasing b is much profitable than decreasing a. When a or b approaches 0, the correct matching becomes 0. Fig. 10 shows that highest correct matching percentage is attained at $b = 0.1$ regardless of the value of a. Therefore, a value of $a = 1$ and $b = 0.1$ will be used as best values of a and b in the proposed similarity measure. The choice of $a = 1$ is to speed up the calculations. Therefore, the proposed similarity measure, for our application, is given by:

$$\mathrm{D}(\mathbf{x},\mathbf{y}) = \sum_{i=1}^{n} |x_i - y_i|^{0.1}. \tag{14}$$

It is worth mentioning here that we stopped at $(b = 0.1)$ because it gives us the highest possible correct matching in our simulation. For other applications, a smaller values of a and b may be used to increase the correct matching percentage.

4.3 Comparison of different similarity measures using MODIS data

In this section, we will compare the proposed similarity measure with the most used similarity measures found in the literature and that have been reported in section 4. The comparison will be done by applying the SOM algorithm for missing values to MODIS data.
Therefore, we will compare different reconstructions of a simulated contaminated version of a MODIS time series of the near infrared channel to its clear version. Due to the difficulty of obtaining a clear MODIS time series on the winter season of the Brittany region in France, we will use a reconstruction version of the time series of 10 dates as yielded in sec. 2.
The procedure of comparison may be described by the schematic representation of Fig. 11. The TS0 in this figure refers to a subset of 401×401 pixels of the original 10 dates time series of near infrared channel that has been used in sec. 2. TS1 in the same figure refers to a reconstruction of TS0 using SOM2 algorithm as in sec. 2. Due to the quantization effect of the Kohonen SOM

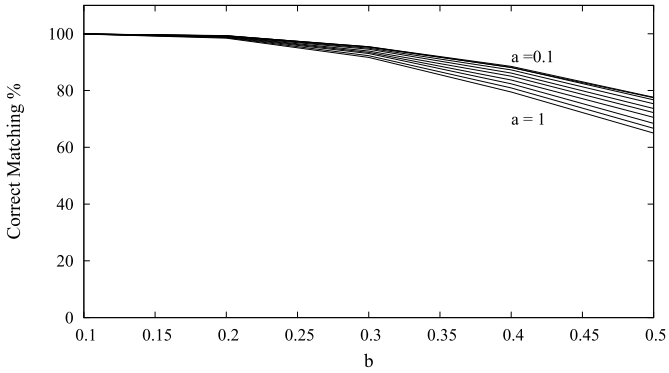
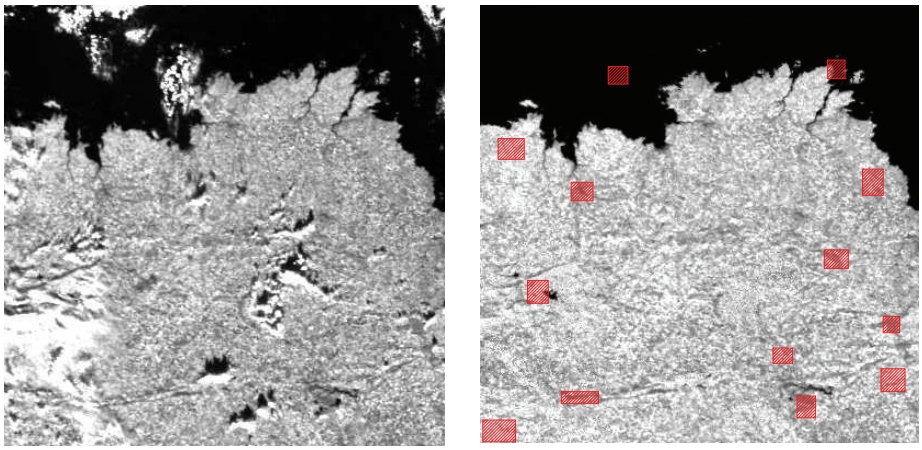


Fig. 10. The percentage of correct matching in function of b while varying a from 0.1 to 1.

algorithm, we will add a random noise in the range between -0.02 and 0.02 reflectance values to the TS1 to produced the time series TS2. The added noise is a random noise drawn from an independent and identical distribution (iid). Images on 01-24-2003 are shown in Fig. 12 for both TS0 and TS1. To simulate clouds and associated shadows, 20% of all pixels in all dates have been selected randomly and have been replaced by a random value between 0 and 0.9 to produce TS3 time series.
In this way a temporal profile of 10 dates, in TS3, has in average two erroneous reflectance values. There are 14375 temporal profiles (from 160801 temporal profiles) that are contaminated in more than 4 dates.



(1) (2)

Fig. 12. Near InfraRed channel of MODIS images on 01-24-2003: (1) the original one and (2) the reconstructed one by the SOM algorithm for missing values.

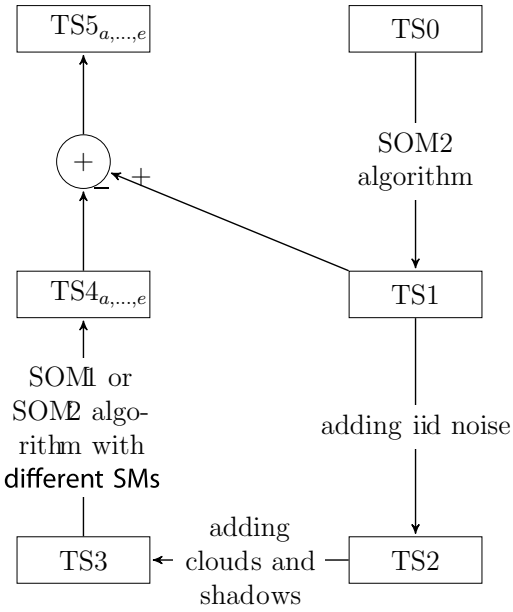


Fig. 11. A schematic representation of steps used to compare the different similarity measures

A comparison of different similarity measures is now possible considering the TS1 as the original time series and a set of the different reconstructions of the MODIS data using different similarity measures($TS4_{a,...,e}$ in Fig. 11).
It remains now to decide what criteria will be used to compare the different reconstructions. We found that statistics of first and second order of a difference time series (difference between original and different reconstructions: $TS5_{a,...,e}$ in Fig. 11) were sufficient to monitor the best similarity measure to be used in such applications. Therefore the average mean and the average standard deviation of each difference time series will be used to find the best similarity measure. An ideal reconstruction will produce a difference image with 0 mean and 0 standard deviation. But due to the fact that the SOM algorithm for missing value has a quantization effect and that 14375 temporal profiles contaminated in 4 dates or more, the 0 mean and standard deviation is not realistic in this comparison. Therefore the nearer the average mean of TS5 to 0 and the smaller the standard deviation are, the better the reconstruction is.

4.3.1 Difference Time Series

In this section we will present results yielded by applying the SOM algorithm for missing values to MODIS data using different similarity measures in the projection phase of the algorithm. Fig. 13 shows the first date of the different reconstructed time series (TS4 in Fig. 11) using the SOM1 algorithm.

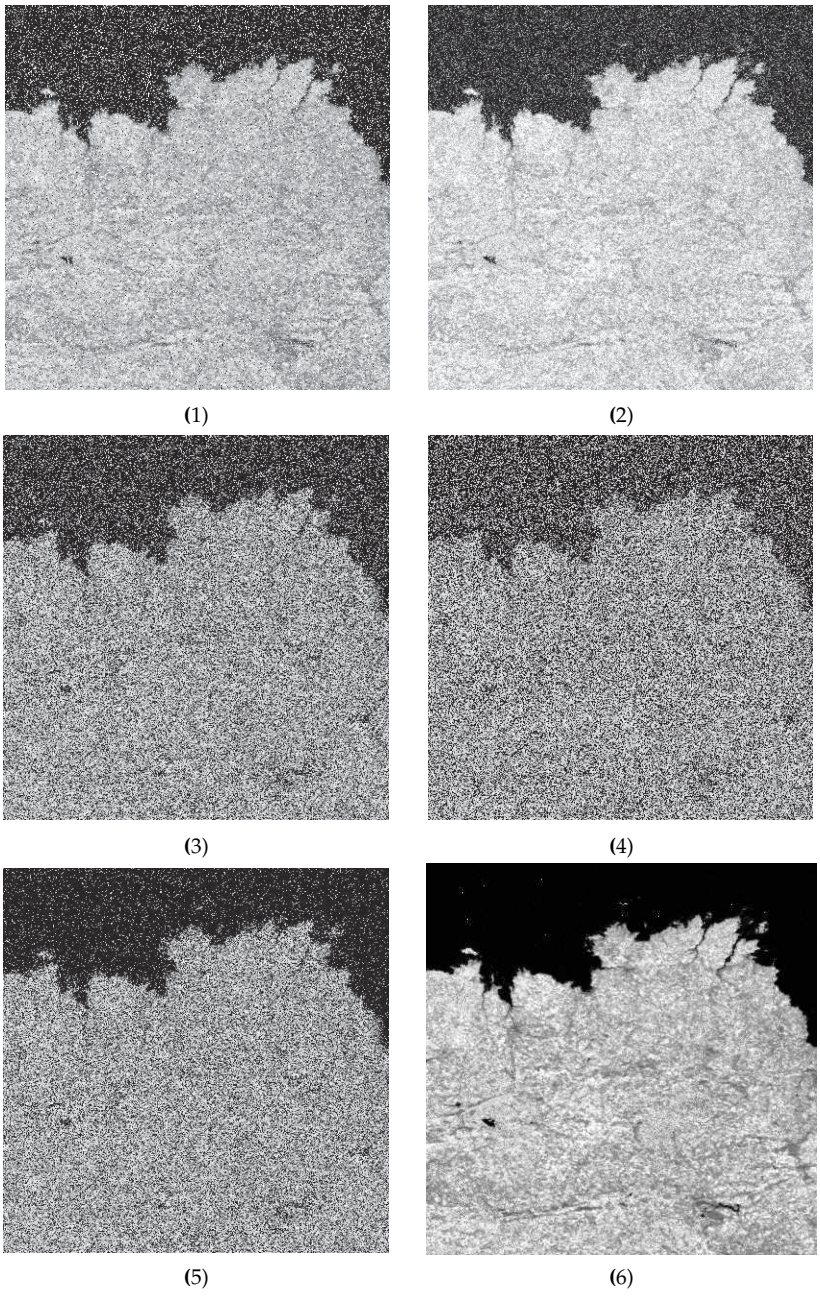
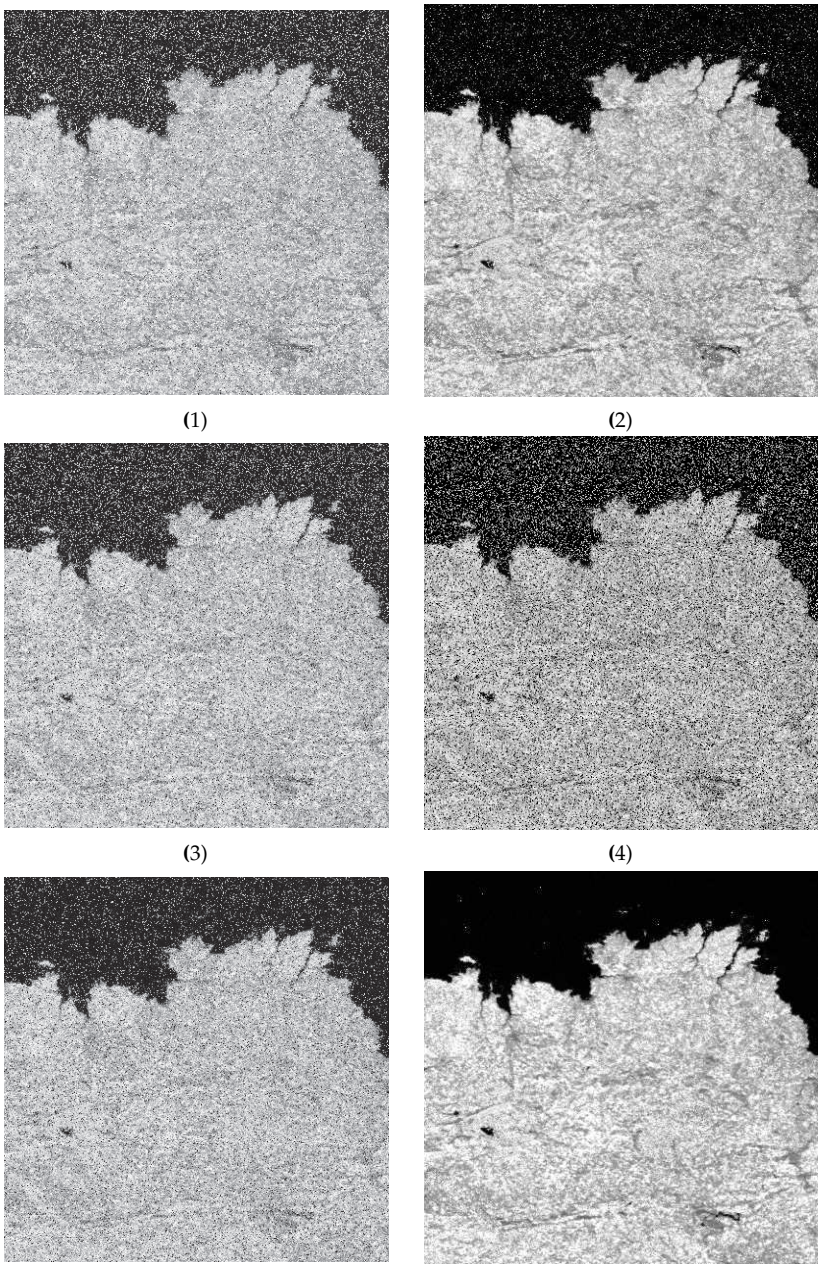


Fig. 13. The simulated cloudy near infrared channel of a MODIS image on 01-24-2003 along with its reconstruction by the SOM1 using different similarity measures: (1) simulated cloudy image, (2) using the Euclidean distance, (3) using the SAM measure, (4) using the SCM, (5) using the SID measure, (6) using the proposed similarity measure.



(1) (2)

(3) (4)

(5) (6)

Fig. 14. The simulated cloudy near infrared channel of a MODIS image on 01-24-2003 along with its reconstruction by the SOM2 algorithm using different similarity measures: (1) simulated cloudy image, (2) using the Euclidean distance, (3) using the SAM measure, (4) using the SCM, (5) using the SID measure, (6) using the proposed similarity measure.

We can notice visually that the near infrared channel of Fig. 13-6 is the only output that may be accepted as a reconstructed version from the one of 12-(2). This image is the one reconstructed by the proposed similarity measure without the need to apply an outlier detector to the contaminated temporal profiles before the projection onto the code vectors.

TS5	(a) ED	(b) SAM	(c) SCM	(d) SID	(e) Proposed SM
mean	-.03575	0.03665	0.03606	0.03878	-0.0002
std	.05594	0.13173	0.14953	0.1218	0.01607

Table 3. Mean and standard deviation of difference images ($TS5_{a,\ldots,e}$) using different similarity measures in the projection phase of the SOM1 algorithm for missing values

Table 3 shows the superiority of the proposed distance measure to all other distance measures used in this experiment. The mean of the time series $TS5_e$, expressed in reflectance value, is two order of magnitude less than all other TS5s. Also the standard deviation is one order of magnitude better than all other TS5s produced by all other similarity measure except the one produced by the Euclidean distance similarity measure.
Fig. 14 shows results using the SOM2 algorithm. We can notice better performance of SOM2 algorithm with respect to the SOM1 algorithm.
Again, the most similar time series to the one in Fig. 12-(2) is 14-(6). It is almost free from the salt and pepper noise contrarily to all other images. The remaining salt and pepper noise, seen clearly in the Pacific ocean, are mainly due to the high contamination of their temporal profiles. Table 4 shows the average of the mean value and the average standard deviation, expressed in reflectance values, of each difference time series ($TS5_{a,\ldots,e}$) using the SOM2 algorithm.
We can also note from the two tables 3 and 4 that the proposed similarity measure gives the best results when using SOM1 or SOM2 algorithm. Moreover, all other similarity measures failed to reconstruct a reliable time series using the SOM1 algorithm. In other words, we have to use SOM2 algorithm when using any similarity measure except with the proposed similarity measure which may be used directly with SOM1 algorithm and still give better results than other similarity measures even with the SOM2 algorithm.

TS5	ED	SAM	SCM	SID	Proposed SM
mean	-0.00736	-0.00233	0.00228	-0.00246	-0.00018
std	0.03516	0.08345	0.09807	0.07718	0.01609

Table 4. Mean and standard deviation of difference images ($TS5_{a,\ldots,e}$) using different similarity measures in the projection phase of SOM algorithm for missing values and without taking into account erroneous components (SOM2 algorithm).

4.3.2 Temporal Profiles

In this section, results yielded from applying the proposed similarity measure and the Euclidean distance to selected temporal profiles are presented in Fig. 15. Each graphic in this figure represents 4 temporal profiles: a real temporal profile (that belongs to TS1), a contaminated temporal profile (that belongs to TS2), a reconstructed temporal profile using SOM1

algorithm along with Euclidean distance similarity measure in the projection phase and a temporal profile yielded from applying the SOM1 algorithm along with the proposed similarity measure in the projection phase.

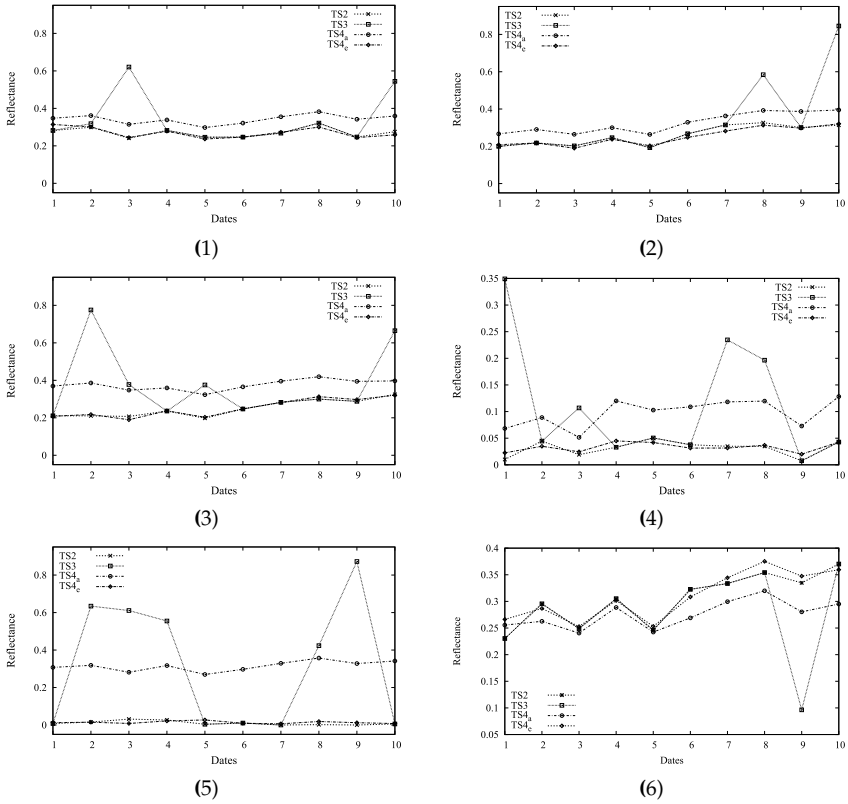


Fig. 15. Examples of matching the temporal profiles using the Euclidean distance and the proposed similarity measure. TS2 represents the original TP, TS3 represents the TP with the added clouds and shadows, $TS4_a$ represents the best matching TP from the SOM using the Euclidean distance and TS_e represents the candidate TP using the proposed similarity measure.

Temporal profiles have been selected to show different possible situations. Fig. 15-(1) and 15-(2) show the reconstruction of a temporal profile which was contaminated by a simulated cloud on two dates at different positions. Fig. 15-(3), 15-(4) and 15-(5) are the same as 15-1 but their original temporal profile was contaminated at four or five dates. The temporal profile in 15-(6) shows another situation where the simulated contamination was a shadow.
We can notice two things: firstly, the proposed similarity measure outperforms the Euclidean distance similarity measure in all situations. Secondly, the Euclidean distance similarity measure is affected by the large differences with respect to small differences and that temporal

profiles reconstructed using the Euclidean distance similarity measure is biased to the direction of contamination (clouds or shadows).

5. Conclusion

A technique to recover erroneous data was presented in this chapter. The application of the technique to temporal series of low resolution images affected by clouds and associated shadows has been studied. This non-parametric algorithm, which is based on the Kohonen's SOM, does not require any statistical models to fit the data. The training phase depends on data issued from the scene to be processed for a certain thematic analysis. The input data is to be considered as reflectance observations with as many spectral bands as allowed by the sensor. Significant results have been found by using a set of 10 images only. Some of these images were very close to each other in time, three of them has two days in-between only. In other words, no uniform sampling is necessary to perform such a temporal profile reconstruction.
In comparison to other methods used to compose time series in order to observe land use and land cover changes, the SOM algorithm along with an outlier detector, SOM2 algorithm, shows several advantages. On the contrary, the MVC has several drawbacks that can be found in the literature Cihlar et al. (1994); Eklundh (1995); Chen et al. (2003) even if it is the most popular technique. In fact, the association of pixels acquired with different zenithal angles and observation conditions creates patchwork artifacts. The MVC is commonly used on NDVI time series and not on reflectance time series whereas the SOM algorithm can be used on reflectance images. Some observations that have no need to be replaced are often replaced with the NDVI maximum value, which is not necessarily the best NDVI value acquired within a period.
A new Similarity measure has been presented to used in the projection phase in the SOM algorithm for missing values. The proposed similarity measure is derived from a non-Gaussian RBF kernel. This proposed similarity measure is performing better than four similarity measures that are widely used in the literature. It performs also with normal distributions as well as with the heavy-tailed distributions.

6. References

Abdel Latif, B., Lecerf, R., Mercier, G. & Hubert-Moy, L. (2008). Preprocessing of Low Resolution Time Series Contaminted by Clouds and Shadows, (7).

Bakar, Z., Mohemad, R., Ahmad, A. & Deris, M. (2006). A comparative study for outlier detection techniques in data mining, *Conference on the Cybernetics and Intelligent Systems*, pp. 1–6.

Chang, C.-I. (2000). An Information-Theoretic Approach to Spectral Variability, Similarity, and Discrimination for Hyperspectral Image Analysis, **46**(5): 1927–1932.

Chapelle, O., Haffner, P. & Vapnik, V. N. (1999). Support Vector Machines for Histogram-Based Image Classification, **10**(5): 1055–1064.

Chen, P., Srinivasan, R., Fedosejevs, G. & Kiniry, J. (2003). Evaluating different NDVI composite techniques using NOAA-14 AVHRR data, *International Journal of Remote Sensing* **24**(17): 3403–3412.

Cihlar, J., Manak, D. & Voisin, N. (1994). AVHRR bidirectional reflectance effects and compositing, *Remote Sensing of Environment* **48**(1): 77–88.

Cottrell, M. & Letrmy, P. (2005). Missing values: processing whith Kohonen algorithm, *International Symposium on Applied Stochastic Models and Data Analysis (ASMDA)*, Brest, France, pp. 17–20.

Eklundh, L. R. (1995). Noise estimation in NOAA AVHRR maximum-value composite NDVI images, *International journal of remote sensing(Print)* **16**(15): 2955–2962.

Fessant, F. & Midenet, S. (2002). Self-Organising Map for Data Imputation and Correction in Surveys, *Neural Computing & Applications* **10**(4): 300–310.

Huang, C., Townshend, J., Liang, S., Kalluri, S. & Defries, R. (2002). Impact of sensor's point spread function on land cover characterization, assessment and deconvolution, *Remote Sensing of Environment* **80**: 203–212.

Ingram, K., Knapp, E. & Robinson, J. (1981). Change-detection technique development for improved urbanised area delineation, *CSC/TM-81/6087*, Report prepared for NASA, Computer Sciences Corporation, Springfield, Maryland.

Jha, C. S. & Unni, N. V. M. (1994). Digital Change Detection of Forest Conversion of a Dry Tropical Forest Region, *International Journal of Remote Sensing* **15**(13): 2543–2552.

Kruse, F. A., Lefkoff, A. B., Boardman, J. W., Heidebrecht, K. B., Shapiro, A. T. & Barloon, P. J.and Goetz, A. F. H. (1993). The spectral image processing system (SIPS)–interactive visualization and analysis of imaging spectrometer data, **44**: 145–163.

Tanre, D., Deroo, C., Duhaut, P., Herman, M. & Morcrette, J. J. (1990). Description of a computer code to simulate the satellite signal in the solar spectrum - The 5S code, *International Journal of Remote Sensing* **11**: 659–668.

van der Meer, F. (2006). The Effectiveness of Spectral Similarity Measures for the Analysis of Hyperspectral Imagery, *Internation Journal of Applied Earth Observation and Geoinformation* **8**: 3–17.

13

Image Clustering and Evaluation on Impact Perforation Test by Self-Organizing Map

Takehiko Ogawa
Takushoku University
Japan

1. Introduction

It is important to estimate the perforation characteristics of materials in the design of the structure that collides with a flying object. Concretely, the ballistic limit velocity and the residual velocity of the material are evaluated in the impact perforation test (Backman & Goldsmith, 1978; Zukas, 1990). Then, the evaluation from the successive images of the perforation process by a super-high-speed camera system has been studied in the mechanical or material engineering fields (Kasano, 1999; Kasano et al., 2001). In this method, a steel ball is shot into the material specimen and the perforation process is taken as successive images. Then, the characteristics of materials are estimated from the location of the steel ball and the behaviour of the specimen. However, the analysis is often difficult because of the scattered fragments. Scattering of the fragments is especially observed in acrylic composites, ceramics and their composite materials which are used in structural applications threatened by high velocity impact. As a result, the accurate evaluation of the characteristic of the material often becomes difficult.

The image clustering is necessary to evaluate the characteristic of the material from the successive images obtained by the impact perforation test. Neural networks are often used as a technique of adaptive image recognition. The neural network can be expected to classify an imperfect image by its robustness. There are a number of neural network models from the viewpoint of the learning method and the weight connection (Peincipe et al., 2000). A number of works have been reported in the pattern recognition by the multi-layered neural network (Ripley, 2007; Ogawa et al., 2006). However, an appropriate selection of learning data and appropriate parameter setting are necessary for data with large variance to use multi-layered neural networks.

The neural network model that performs unsupervised and self-organizing learning is effective to such a difficult problem (Kohonen, 1989; Kohonen, 2001). It can classify the images based on the competitive learning. The self-organizing map has been studied as an unsupervised image clustering method. Because the self-organizing map can classify the image based on the unsupervised and self-organizing learning, it is not necessary to prepare any learning data for a number of fragment patterns. We proposed to apply the self-organizing map to the image classification in the impact perforation images. Concretely, we

classify a steel ball, background, and fragments in the impact perforation images by the self-organizing map (Horiuchi et al., 2004).
The self-organizing map performs the mapping from the input data to competitive neurons while maintaining the interrelation among the input data. The self-organizing map is suitable for the image clustering because it can express the order and the similarity between the provided data. Therefore, the distance between patterns can be estimated from the mapping result, because the self-organizing map realizes the mapping that reflects the topological relation. That is, the quantitative evaluation of the difficulty of the image clustering becomes possible. We also propose to apply the self-organizing map to evaluate the difficulty of the image clustering in the impact perforation images. Concretely, we evaluate the difficulty of the impact perforation images of each composite material by the self-organizing map (Okubo et al., 2007). Moreover, the effect of the self-organizing map is confirmed by the simulation.
In this chapter, I show the effectiveness of the self-organizing map for the image clustering of actual composite materials; polycarbonate (PC), polymethyl methacrylate (PMMA) and alumina (Al_2O_3). Also, the difficulty of the clustering of the given image is quantitatively evaluated.

2. Impact Perforation Test

In the impact perforation test, a small steel ball is shot into the material specimen to examine the perforation and destruction process. The system for the impact perforation test with a super-high-speed camera is used to observe the perforation process. In this system, we can observe the states before and after the impact perforation by successive images.
When the steel ball perforates through a plate, the residual velocity of the steel ball after the perforation is expressed by several characteristics; the strength of the board, rigidity, initial form, size and so on. For example, we can estimate the material property according to the impact velocity, the residual velocity of the steel ball, and the geometric property of the material. Kasano et al. have been studying the evaluation of the material property by the impact perforation test based on the above-mentioned principle. The residual velocity V_R is expressed by

$$V_R = F(a_1, a_2; V_i) \tag{1}$$

where a_1, a_2 and V_i mean the material properties, geometric properties of the material and initial velocity of a steel ball. We can estimate the material property a_1, if we know the initial velocity V_i, residual velocity V_R of the steel ball and the geometric property a_2. The perforation limit velocity is one of the important material properties, and can be estimated by the initial velocity and the residual velocity that are obtained in the impact perforation test (Kasano et al., 2001).
As one of the measuring method of the velocity of a steel ball, a super high-speed camera is used. In this system, the steel ball launched from the shooting device perforates through a monotonous material plate in the high temperature furnace, and the appearance is taken of a picture with the high-speed camera. The high-speed camera of this system can take a picture of four successive images. The experimental setup that we used is shown in Fig.1. We can measure the velocity of the steel ball with the high-speed camera from the

successive images of the perforation. However, the location of the steel ball cannot often be measured in precision by the fragments of the destructed material in the perforation image of an actual material. In the current system, the image of the impact perforation test is visually classified. The classification from the successive images is often difficult because of the fragments of the material. Therefore, precise image clustering of a steel ball, background and fragments is necessary. We propose the use of neural network to classify them in the image with degradation by fragments of the specimen. We show the image classification of the images degraded by the scattered material fragments accurately by using the neural network (Ogawa et al., 2003).

The successive images of the impact perforation test with actual composite materials; polycarbonate (PC), polymethyl methacrylate (PMMA) and alumina (Al_2O_3) specimens are shown in Fig.2. The number drawn in the figure shows the photographed order. The image of PC specimen is so clear that we can visually classify it, sufficiently. However, the classification of the image of PMMA specimen is a little more difficult than that of PC. Moreover, the classification of the image of alumina is too difficult to classify because of scattered fragments. The aim of this study is to classify these images accurately. The plate size, thickness of the specimen, the impact velocity of the steel ball and the interframe time of successive images are shown in Table 1. The size of the steel ball was 5mm in diameter, and 0.5g in mass.

The polycarbonate (PC) has the transparency equal with the glass and the highest impact-proof in plastic, and is used for the consumer electronic and the mobile device and so on. The impact-proof of PC equals a metallic material. However, there is a fault of weakness to an alkaline medicine and an organic solvent. The polymethyl methacrylate (PMMA) has the highest scratch-proof and transparency in transparent plastic, and is used for the window, lens and the housing of various equipments. However, it is a little inferior to the impact-proof, and is a polymeric material that causes the same brittleness destruction as ceramics. Because the alumina is light and has the corrosion-proof, it is expected as a material of the machine structure in the ultra high temperature. However, there is a fault that resistance to the mechanical shock and thermal shock is extremely low because the destruction toughness is low and fragile. The impact perforation images of these three kinds of specimens are classified into the steel ball, the background, and the fragments (material).

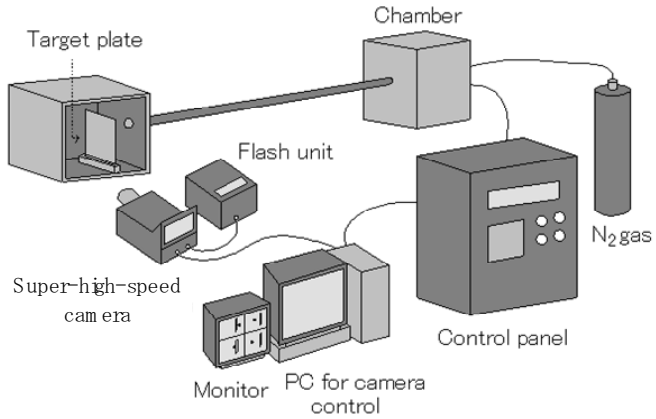


Fig. 1. Experimental setup of the impact perforation test.

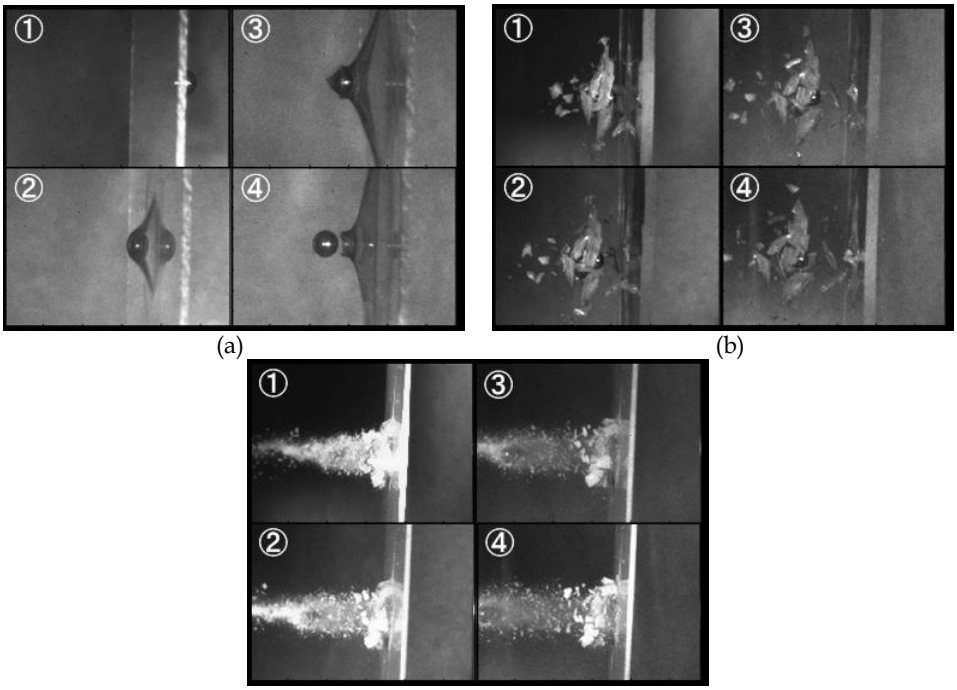


Fig. 2. Successive images obtained by super high-speed camera in the impact perforation test with composite materials, (a) polycarbonate (PC) specimen, (b) polymethyl methacrylate (PMMA) specimen and (c) alumina specimen.

Material	PC	PMMA	alumina
Image size (pixel)	1172x770	1000x656	1172x770
Specimen's size (mm)	100x100	80x60	80x80
Specimen's thickness (mm)	1.5	3.0	1.5
Initial velocities of a steel ball (m/s)	220	309	224.6
Interval time of successive images(μs)	50	150	50

Table 1. Parameters of the impact perforation test and images.

3. Self-Organizing Map

There are a number of supervised learning models and unsupervised learning models in the neural network. Typical examples of the former and the latter are the multi-layered neural network and the self-organizing map, respectively. The multi-layered neural network learns the input-output relation using the error back-propagation method. Because the number of learning data are limited when the input-output mapping is learned by the multi-layered neural network, the network supplements the answer of the problem which was not provided as the learning data with the generalization ability. The multi-layered neural

network has the problem of over-learning in which the learning progresses only for the provided learning data and the generalization ability decreases (Ogawa, 1992).
On the other hand, the self-organizing map is an unsupervised learning model of neural network, which is based on the competitive learning method. Typical self-organizing map is composed of an input layer, a competitive layer and interlayer connection weights, as shown in Fig.3. An input neuron has the connection weights with all competitive neurons. Since the learning expresses the interrelation of input data, a competitive layer forms the mapping that reflects the distributions of input data. In the self-organizing map, we need not provide to which category input data belongs, and input data are classified by self-organizing. The interrelation among the input data is mapped to the neurons arranged in one or two-dimensional area. Therefore, it is considered that the problem of supervized learning model can be overcome.
At the learning phase of the network, the neuron that responds most strongly becomes a winner while a lot of neurons respond for a given input, and the connection weights of the winner neuron and the neighborhood neurons are updated. The learning is done for not only the winner of the competition but also their neighborhoods. In a word, the adjacent neurons have mutually similar connection weights because they learn the similar data. The template that reflects a statistical distribution of data and mutual analogous relationship is formed by such learning. Therefore, the similarity of the input data can be measured according to the relative distance of them. The concrete procedure of competitive learning of the self-organizing map is as follows. First, it identifies a winner neuron. Then, the weights of the winner neuron and the neighborhoods are updated by the input vector in each learning step. The weights of the winner neuron and neighborhood are updated in proportion to the learning rate. The self-organizing map is suitable for clustering that classifies the data set into some groups without learning data, because it is able to express the similarity of the data set.
In this study we propose to use the self-organizing map for classifying the impact perforation images. The self-organizing map is used to distinguish a steel ball, background and fragments. In the impact perforation images, the shapes of the fragments are different on each material, experimental condition and so on. The image in a steel ball is also slightly different according to differences of the quantity of light and so on. So, it is difficult to prepare the appropriate and sufficient learning data for the material specimen and the experiment environment, beforehand. We use the self-organizing map which is one of the competitive learning type neural networks. As mentioned above, the unsupervised learning of the self-organizing map is not affected by the quality of the learning data. The impact perforation image is classified by the self-organizing map. The self-organizing map recognizes the category by a partial space which the data belongs to the division of the category of the feature vector space, where data exists with the represented template.

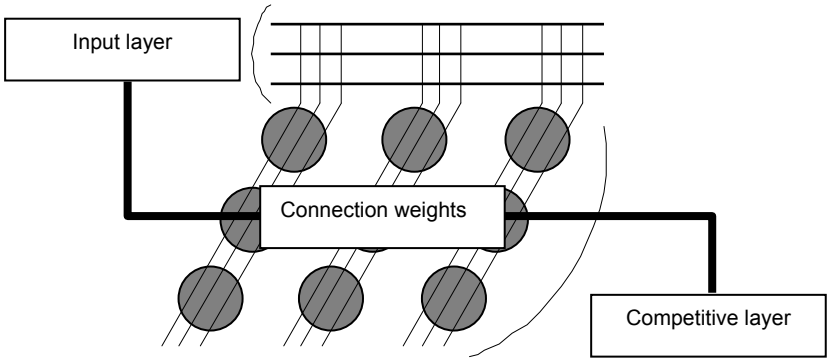


Fig. 3. Typical architecture of self-organizing map.

3.1 Unsupervised Learning of Self-Organizing Map

In the self-organizing map, a number of neurons fire to the given input. The neuron with the strongest output becomes a winner, and the connection weights are updated to the winner neuron and its neighbourhoods. The input neurons and the competitive neurons are connected in parallel between all neurons through the connection weights. In general, these weights have different values for different neurons. The output values of the competitive neuron to a certain input vector are compared by all competitive neurons, and the position that reaches the highest value in a certain measure is considered to be the response position. Here, we consider the network with M input neurons and N competitive neurons. The connection weights vector from the i-th input neuron and a certain input vector are assumed to be $w_i=(w_{i1}, w_{i2}, ..., w_{iN})$, i=1, 2, ..., M and $x=(x_1, x_2, ..., x_M)$. When input data are given, we compare the distances between the input vector and each competitive neuron to dicide the winner neuron k. The distance D_j on j-th competitive neuron is defined by euclid distance as

$$D_j = \sum_{i=1}^{M} \left| x_i - w_{ij} \right| \tag{2}$$

where M means the number of input neurons. Based on the competitive learning, the output of the neuron Z_k with the most similar weight becomes one, while the output of the other neuron become zero. Then, the weights of the winner neuron and its neighborhoods N_k are updated. The weights of the winner neuron from the i-th input neuron are updated as

$$w_{ik}(t+1) = w_{ik}(t) + \alpha(t)(x_i - w_{ik}(t)) \tag{3}$$

where $\alpha(t)$ means the learning coefficient that is reduced as the learning epoch. In general, the weights of the neighborhood neuron is corrected at the update rate that is lower than that of the winner neuron. In this study, the weights of the neighborhood neurons are updated as

$$w_{ik}(t+1) = w_{ik}(t) + \frac{\alpha(t)}{2}(x_i - w_{ik}(t)) \tag{4}$$

The other neurons are not corrected as

$$w_{ik}(t+1) = w_{ik}(t) \tag{5}$$

The range of the correction of the weights is shown like Fig.4. The weight w_{ik} of the winner neuron k is corrected in the direction of the input vector x_i, and approaches to the input vector x_i. Parameter $\alpha(t)$ is a learning coefficient, and is the parameter that adjusts how to approach it to x_i. Variable t means the epoch number, and the set $N_c(t)$ of the neighborhood neurons of winner k and the learning coeffiecinet $\alpha(t)$ are the function of t. Learning coefficient $\alpha(t)$ is initially set to the comparatively large value, and is decreased gradually with the learning epoch t. Also, the size of the neighborhood region $N_c(t)$ is decreased with the learning epoch. All the weights of the neuron move greatly toward the input area where the input vector exists. Then, the map orders itself to the given input vector, as the neighborhood distance decreases. The relation between the weight correction and the learning coefficient is shown in Fig.5.

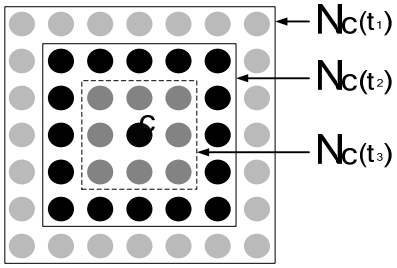
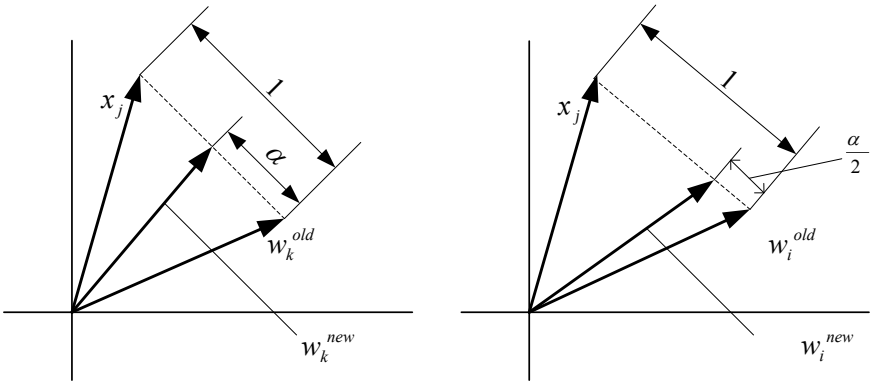


Fig. 4. Winner neuron and its neighbourhood neurons.



(a) (b)

Fig. 5. Relation between weight correction and learning coefficient, (a) winner neuron (b) neighborhood neurons.

4. Image Clustering and Evaluation by Self-Organizing Map

In this section, the architecture and the procedure of the self-organizing map for image clustering and evaluation on the impact perforation test are explained. The self-organizing map used in this study has the one-dimensional competitive layer. The input is the one-dimensional vector by which the feature of the image is extracted, and the output is a mapped vector which appears on the one-dimensional competitive layer. The purpose of the image clustering is the classification of a steel ball, background and fragments in the impact perforation images. Also, the purpose of the image evaluation is to evaluate the difficulty of the image clustering from the output distribution on the competitive layer. The network architecture used in this study is shown in Fig.6.
The network is used in the three steps; the learning, the classification, and the evaluation. They are explained by the following subsections.

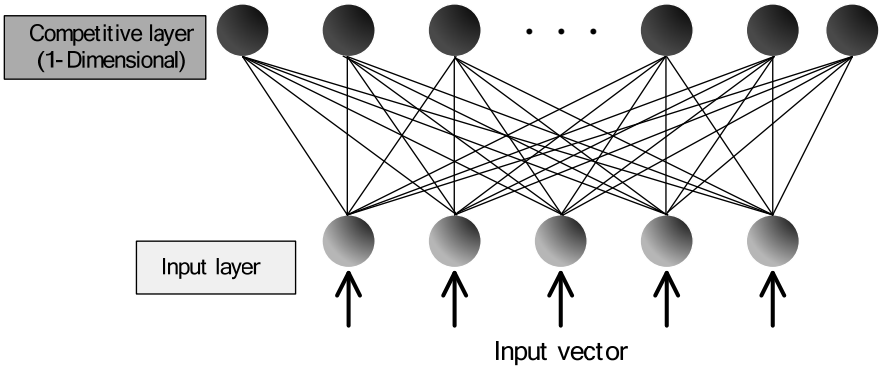


Fig. 6. Network architecture

4.1 Learning by Self-Organizing Map

Here, the learning method of the self-organizing map is explained. In general, we compose the input vector after compressing the image, and classify it in the image classification. In this study, the sub-images of the steel ball, the background, and the fragments are extracted from the original images. We use two sub-images of specimen fragments, because the fragments of the specimen is various shape. The quantitative features are calculated from those sub-images as input vectors. The mapping forms the template of the steel ball, the background, and the fragments of the specimen by unsupervised learning of the input vector obtained from these learning sub-images.
The network first identifies the winner neuron for the given input vector. Then, the vector expressed by the weights of the winner neuron and those of the neighborhood neurons are brought close to the value of the input vector in each learning step. The weights of the winner neuron are changed for proportion to the learning rate, while those of the neighborhood neuron are changed for proportion to the half of the learning rate. The learning rate and the neighborhood distance of the winner neuron are updated through two phases of the ordering phase and the tuning phase. To use these two phases, the following four are defined (Demuth et al., 2009).

Ordering phase

Ordering phase lasts for the given number of steps. The neighborhood distance starts at a given initial distance, and decreases to the tuning neighborhood distance. As the neighborhood distance decreases over this phase, the neurons of the network typically order themselves in the input space with the same topology in which they are ordered physically.

Tuning phase

Tuning phase lasts for the rest of learning or adaptation. The neighborhood distance stays at the tuning neighborhood distance, which should include only close neighbors, i.e., typically 1.0. The small neighborhood fine-tunes the network, while keeping the ordering learned in the previous phase stable.

As with competitive layers, the neurons of a self-organizing map will order themselves with approximately equal distances between them if input vectors appear with even probability throughout a section of the input space. If input vectors occur with varying frequency throughout the input space, the feature map layer tends to allocate neurons to an area in proportion to the frequency of input vectors there. Thus, feature maps, while learning to categorize their input, also learn both the topology and distribution of their input. The summary of the procedure is as follows.

Step 1. The sub-images of the steel ball, the background, and fragments of specimen are extracted from the impact perforation images as a learning image.

Step 2. The input vector by quantitative features is calculated from the extracted sub-images.

Step 3. The calculated input vector is input to the network to update the weights.

4.2 Image Clustering by Self-Organizing Map

Image clustering by the self-organizing map is performed by the learned network. The learning images are mapped to the competitive layer. While the output of only the nearest neuron in the map becomes 1.0 if the input data of the image classified into the network is provided, those of other neurons become 0.0. The procedure of clustering is as follows.

Step 1. The labeling is done to the neuron in the competitive layer after the learning.

Step 2. The images to be classified is divided into the decided size, and the input vector is calculated.

Step 3. After the calculated input vector is input to the network, and the network outputs it from the result of the response and the labeling in the competitive layer.

To classify the input image, it is necessary to confirm which neuron in the competitive layer represents the input data of the learning image. We input each learning image and label the responding competitive neurons. As a result, the labeling is done to the neuron in the competitive layer and the image is classified. Figure 7 shows the concept of the labeling. The concept of the network architecture in the image clustering is shown in Figure 8.

The classification images are four successive images, and the classication is performed to each image. The classification images are scanned and cut out to the sub-images. After the calculation of the input vector for each sub-image, it is classified with the label of the neuron that responds to the input vector.

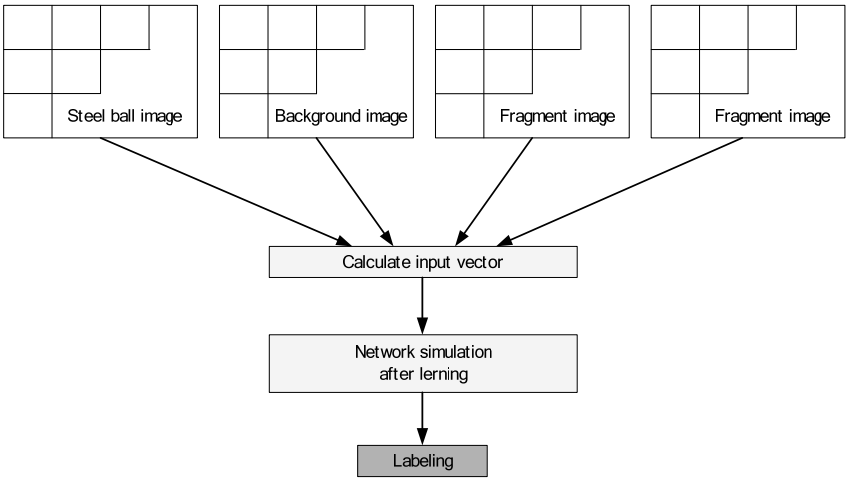


Fig. 7. Concept of labeling

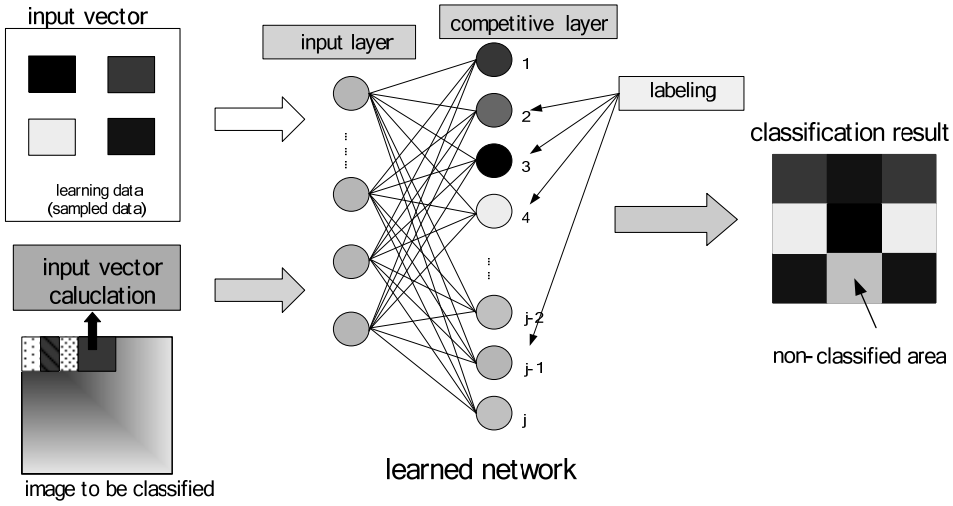


Fig. 8. Concept of the network architecture in image clustering

4.3 Image Evaluation by Self-Organizing Map

The self-organizing map is able to map the input to the competitive layer with the topological relationship between the given input vectors. In a word, the topological relationship can be estimated from the distribution expressed on the competitive layer neuron. Based on this feature, we proposes the method of quantitatively evaluating the difficulty of the classification of the given image by using the self-organizing map for the topological relationship between images of the steel ball, the background, and the fragments of specimen.

The steel ball, the background, and the fragments are labeled at the same time as the image's being classified by the self-organizing map. The location where each neuron responds to the output distribution on the competitive layer can be shown from the result of the labeling. Then, the location of fired neuron on the competitive layer in each pattern is confirmed. Figure 9 shows the concept of the image evaluation.

The difficulty and accuracy of the image classification can be estimated from the location of the firing neuron on the competitive layer for each pattern. For instance, if each pattern of the steel ball, the background, and fragments is classified by the firing neuron position of a competitive layer in each pattern clearly, the image can be judged to be an easy classification. However, it can be judged that the image is difficult classification if the firing neuron position by each pattern is overlapping. Also, it is similarly used as an evaluation method of the classification result. If each pattern is separate, the image is classified almost accurately. On the other hand, if each pattern overlaps, it is appreciable with an inaccurate image classification.

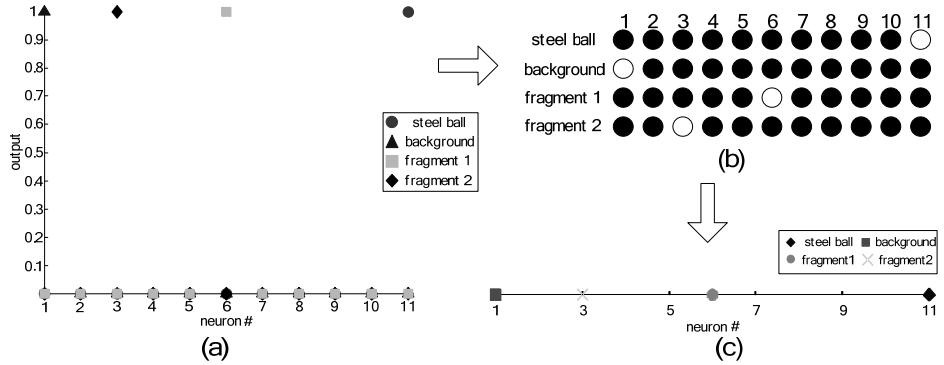


Fig. 9. Concept of image evaluation by self-organizing map, (a) result of labeling, (b) situation of firing on competitive layer, and (c) location of firing neuron on competitive layer for each pattern.

4.4 Input Vector of Self-Organizing Map

In this study, two kinds of input vector are examined to classify the image. The first method is to calculate the quantitative features from the sub-image and to compose the input vector. First of all, the sub-image is extracted from the impact perforation image. Then, the quantitative features; the standard deviation, the difference of the maximum value and minimum value, the median value, the maximum frequency value, the mean values of the four corners, the maximum values, the minimum value and the mean value in the pixels. The one used for the input vector calculation was selected from these quantitative features by the trial and error.

Next, the method of inputting immediate pixel values is examined. The sub-image is extracted from the impact perforation image as well as the input vector by the amount of characteristic. Then, the pixel values of five points are extracted from the sub-image. The input vectors used in the simulation are shown in Table 2.

Input vector	Quantitative features	Pixel values
PC	Standard deviation Difference of max and min values Maximum value Minimum value	Five pixel values
PMMA	Difference of max and min values Median value Maximum frequency value Maximum value Minimum value Mean value	
Alumina	Standard deviation Difference of max and min values Median value Maximum frequency value Maximum value Minimum value Mean value	

Table 2. Input vectors used in the simulation.

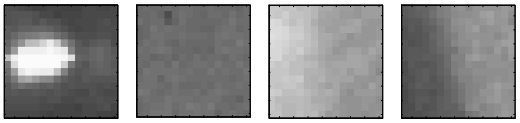
5. Simulation

In this study, we use the impact perforation images of PC (polycarbonate), PMMA (polymethyl methacrylate) and alumina, shown in Fig.2. Each image consists of four successive photos. The parameters are shown in Table 1. The PC image can be visually classified. The classification of the PMMA image is comparatively easy though it is not clearer than the PC image. However, it is difficult to classify the alumina image because of the scattering of the fragments. In this study, these images are classified by the image recognition network using the self-organizing map. In addition, the quantitative evaluation of the difficulty of image classification by the self-organizing map is examined.

The learning image and the classification image are extracted from the original impact perforation images. As for the learning image of PC, the steel ball, the background and two material images of 16x16 pixels are extracted from the second image. The learning image of PMMA is extracted from the third image in the same size as PC. As for the learning image of the alumina image, the steel ball, the background of 16x16 pixels and two fragment images of 48x48 pixels are extracted from the third image. These learning images of PC, PMMA and alumina are shown in Figs. 10, 11 and 12 respectively. Then, the classification images are extracted from each original image (PMMA, PC and alumina). The extracted images are shown in Fig. 11. The size of each classification image is 256x256. These four images are classified into four kinds; steel ball, background and two fragments.

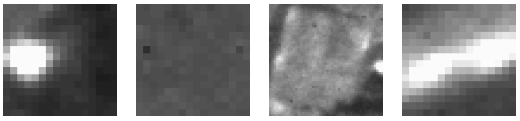
To examine two kinds of input values in which the quantitative features and the immediate pixel values of the learning image, we input these to the network in learning phase. The classification simulation is carried out by two methods of the quantitative features and the immediate pixel values, and the difficulty of the image and the accuracy of the classification are evaluated from the output distribution of the competitive layer.

The network parameters are shown in Table 3. The network parameters are changed according to the input vector and the classified image. In the case of the quantitative features, we decided to adopt the features by trial and error. On the other hand, we sampled five pixels as the immediate pixel value input. We used 11 neurons that were arranged in one dimension as a competitive layer.



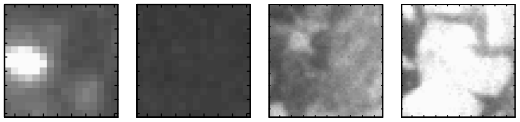
(a) (b) (c) (d)

Fig.10 Learning image of PC, (a)steel ball, (b)background, (c) fragment1, and (d)fragment2



(a) (b) (c) (d)

Fig. 11. Learning image of PMMA, (a)steel ball, (b)background, (c) fragment1, and (d)fragment2



(a) (b) (c) (d)

Fig. 12. Learning image of alumina, (a)steel ball, (b)background, (c) fragment1, and (d)fragment2

Material	PC	PMMA	alumina
Input vector	4 features or 5 pixel values	6 features or 5 pixel values	7 features or 5 pixel values
Number of input neurons	4 or 5	6 or 5	7 or 5
Number of competitive neurons	11		
Ordering phase step number	1000		
Ordering phase learning rate	0.9		
Tuning phase step number	0.01		
Tuning phase neighbourhood distance	1		
Learning epoch number	500		

Table 3. Network parameters.

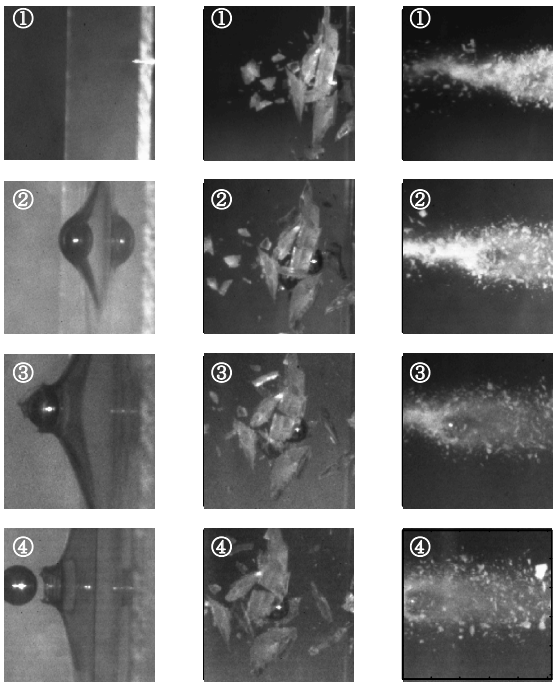


Fig. 13. Classification images, (a) PC, (b) PMMA and (c) alumina

5.1 Results

The simulation was carried out to confirm the classification and the quantitive evaluation of the difficulty of the classification by the amount of characteristic and pixel value input of PC, PMMA and alumina impact perforation image.

First, the classification result of PC is shown in Fig. 14. The white area in the classification result shows the steel ball. The gray area shows the background and fragments. A black area means unclassificated area. According to the result, the place of the steel ball can be distinguished. Neither the background nor the material are distinguished accurately partially. The output distribution of a competitive layer for each patterns is shown in Fig.15. In Fig.15, the white circle and the black circle mean the acted and not acted neurons, respectively. From result of Fig.15, it is understood that each pattern is separate to each learning image clearly in the PC image. Because the distance between patterns of the steel ball and the material is considerably large, it is considered that the distinction is the easiest.

Next, the classification result of PMMA is shown in Fig.16. The result is shown similar to the case of PC. From the result, we can see to distinguish the steel ball clearly. Moreover, because the distinction of the background and the fragment is easy, it is possible to classify it. The output distribution of a competitive layer is shown in Fig.17. As for the PMMA image, each pattern is separate to each learning image clearly. Because the distance between a steel ball and fragment patterns is large, it is considered that the distinction becomes easy.

Finally, the classification result of alumina is shown in Fig.18. From the result, it is difficult to distinguish the steel ball because of the scattered fragments. However, the fragments and the background is distinguished correctly. The output distribution of a competitive layer is shown in Fig.19. In the alumina image, it is considered that each pattern adjoins each learning image. Because the distance between a steel ball and fragment patterns is small, it is considered that the distinction becomes difficult.

According to the above-mentioned result, the classification of the steel ball that is necessary to estimate the strength property of the material was sufficiently performed in three impact perforation images. Also, the classification of the fragments that is important to estimate the destruction property was almost done, too. Moreover, it was shown that the distinction of the steel ball was easy in order of the PC image, the PMMA image, and the alumina image according to the result of the evaluation by the output distribution in a competitive layer. From the results of Figs. 15, 17 and 19, the mean distances between the firing positions to the steel ball and the fragments on the competitive layer are calculated and are shown in Table 4. These values are almost corresponding to an intuitive difficulty to classify each image. That is, it was clarified that the output distribution of the competitive layer expressed the difficulty of the quantitative evaluation of the classification difficulty by the simulation. Especially, it was confirmed that the distance between the competitive layer neurons that responded to the input vector of the steel ball and the fragments expressed the classification difficulty. Therefore, the difficulty of the image classification can be quantitively evaluated by measuring the positions of the responding competitive layer neurons.

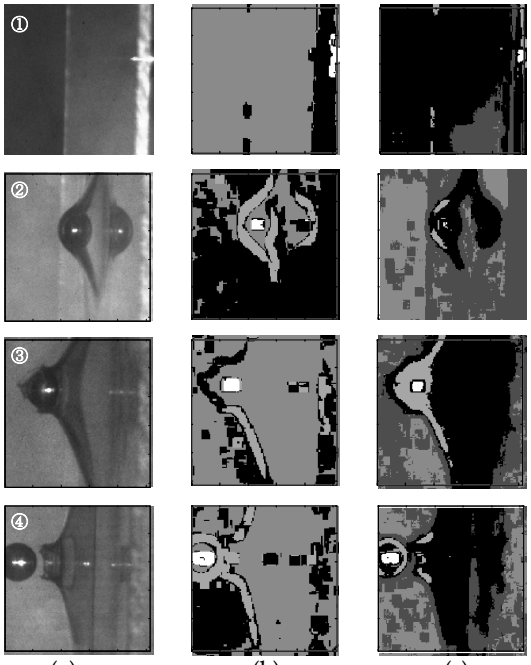


Fig. 14. Classification result for PC specimen image. (a) original image, (b) result for quantitative feature input, (c) result for pixel value input.

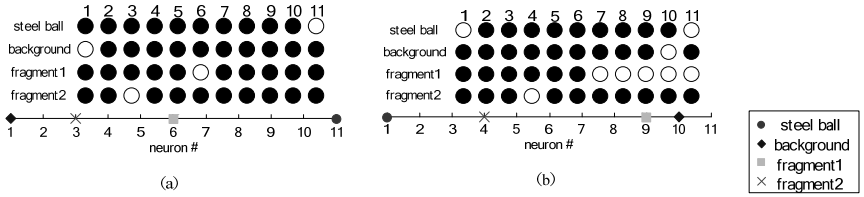
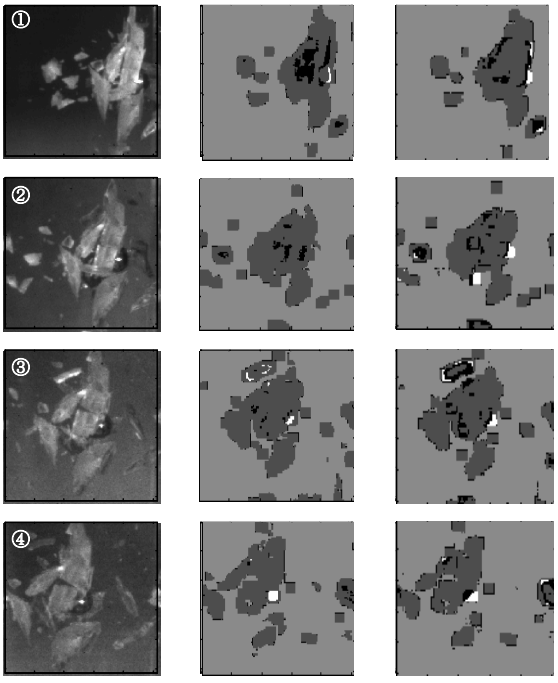


Fig. 15 Output distribution of a competitive layer for PC specimen image. (a) result for quantitative feature input and (b) result for pixel value input.



(a) (b) (c)

Fig. 16. Classification result for PMMA specimen image. (a) original image, (b) result for quantitative feature input, (c) result for pixel value input.

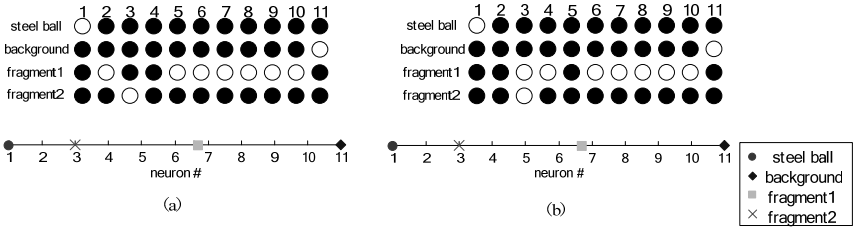


Fig. 17. Output distribution of a competitive layer for PMMA specimen image. (a) result for quantitative feature input and (b) result for pixel value input.

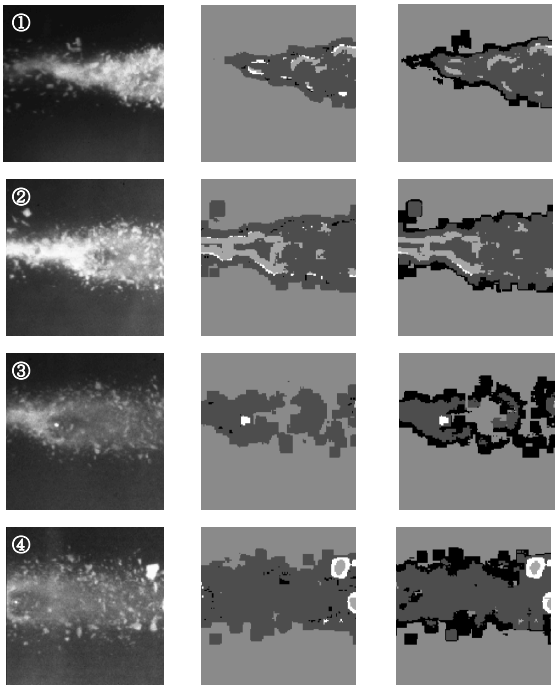


Fig. 18. Classification result for alumina specimen image. (a) original image, (b) result for quantitative feature input, (c) result for pixel value input.

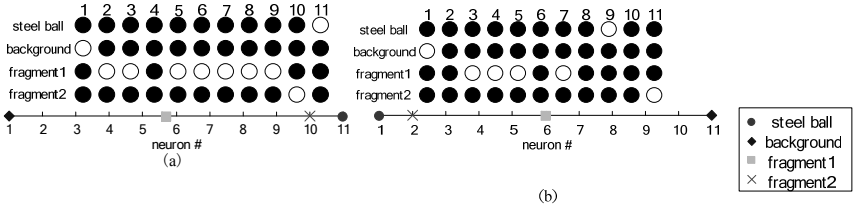


Fig. 19. Output distribution of a competitive layer for alumina specimen image. (a) result for quantitative feature input and (b) result for pixel value input.

Input \ Materials	PC	PMMA	alumina
Quantitative features	6.5	3.6	3.1
Pixel values	5.5	3.9	3.1

Table 4. Distances between firing neurons corresponding to a steel ball and fragments.

6. Conclusion

In this study, we proposed to use the self-organizing map for the classification of the impact perforation image, as one of the image recognition problems by the neural network. Then, the classifications of three impact perforation images (PC, PMMA and alumina specimens)

were used to examine the image recognition by the self-organizing map. The quantitative features and the immediate pixel values of the images were used and simulated as the network inputs, and the effect of the self-organizing map was confirmed. Since the self-organizing map maintained the topological order in the data space, it was confirmed to be able to judge the difficulty of the image classification from the output result.
As for the PC image, the steel ball was clearly distinguished but the material was not classified sufficiently. The reason is considered that the self-organizing map was not able to extract the feature of the image accurately. In the PMMA image, the steel ball and the fragments were classified clearly. The reason is considered that the self-organizing map extracted the features of the image almost accurately. In the alumina image, the steel ball was not classified sufficiently, because of the scattered fragments. However, the fragments and the background were separated clearly. Moreover, the difficulty of the image classification was evaluated from the output distribution on the competitive layer neuron to each pattern. Concretely, the reason why the classification of the steel ball in the alumina image is difficult was shown though that in the PC and PMMA images was easy.
As the future problems, the improvement of the classification accuracy and the evaluation accuracy is considered. First of all, we have to examine the input vector calculation method to extract more features for more accurate classification. It is necessary to examine the vector calculation method that extracts the features of the image, while the two types of the input vector; the quantitative features and the immediate pixel values were used in this study. Moreover, it is necessary to examine the relation between the accuracy and the composition of the competitive layer for the quantitative evaluation of the difficulty of the image classification. It is considered that there is an influence on the classification accuracy by increasing the number of competitive layer neurons to improve the evaluation accuracy.

7. References

Backman, M. E. & Goldsmith, W. (1978). The Mechanics of Penetration of Projectiles into Targets, *International Journal of Engineering Science*, vol. 16, issue 1, pp.1-99.

Demuth, H.; Beale, M. & Hagan, M. (2009). *Neural Network Toolbox™ 6 User's Guide*, The MathWorks, Inc.

Horiuchi, T.; Ogawa, T. & Kanada, H. (2004). Analysis of Impact Perforation Images Using Self-Organizing Map, *Proceeding of the SICE Annual Conference*, pp.605-608.

Kasano, H. (1999). Recent Advances in High-Velocity Impact Perforation of Fibber Composite Laminates, *JSME International Journal A*, Vol. 42-2, pp.147-157.

Kasano, H. ; Okubo, T. & Hasegawa, O. (2001). Impact Perforation Characteristics of Carbon/Carbon Composite Laminates, *International Journal of Materials and Product Technology*, Vol.16, No.1-3, pp.165-170.

Kohonen, T. (1989). Self-Organization and Associative Memory, Springer-Verlag, ISBN: 978-3540183143, Berlin.

Kohonen, T. (2001). Self-Organizing Maps, Springer-Verlag, ISBN: 978-3540679219, Berlin.

Ogawa, H. (1992). Neural Networks Learning Generalization and Over-Learning, *Proceeding of the International Conference on Intelligent Information Processing & System*, vol. 2, pp. 1-6.

Ogawa, T.; Kanada, H. & Kasano, H. (2003). Neural Network Localization of a Steel Ball in Impact Perforation Images, *Proceeding of the SICE Annual Conference*, pp.416-419.

Ogawa, T. ; Tanaka, S. ; Kanada, H. & Kasano, H. (2006). Impact Perforation Image Processing Using a Neural Network, *Proceeeding of the SICE-ICASE International Joint Conference*. pp. 3762 – 3765.

Okubo, K. ; Ogawa, T. & Kanada, H. (2007). Impact Perforation Image Processing Using a Self-Organizing Map, *Proceeding of the SICE Annual Conference*, pp.1099-1103.

Principe, J. C.; Euliano, N. R. & Lefebvre, W. C. (2000). *Neural and Adaptive Systems*, John Wiley & Sons, ISBN: 0-471-35167-9, New York.

Ripley, B. D. (2007). *Pattern Recognition and Neural Networks*, Cambridge University Press, ISBN: 978-0521717700, New York.

Zukas, J. A. (1990). *High Velocity Impact Dynamics*, John Wiley & Sons, ISBN: 978-0471514442, New York.

14

Self-Organizing Map-based Applications in Remote Sensing

Anthony Filippi[1], Iliyana Dobreva[1],
Andrew G. Klein[1] and John R. Jensen[2]
[1]*Department of Geography, Texas A&M University*
[2]*Department of Geography, University of South Carolina*
USA

1. Introduction

Remote sensing involves the collection of information about an object from a distance. Often remote-sensing instruments are mounted onboard an air- or space-borne platform and typically record electromagnetic energy in specific wavelength intervals, or bands. The electromagnetic energy recorded over a given area contains information about surfaces reflecting or emitting energy. This information can be used for a variety of applications; for example, remote-sensing image analysis can extract thematic information such as land-cover types (Jensen, 2005).

Artificial neural network (ANN) techniques have increasingly been employed in the analysis of remotely-sensed images. ANNs can be advantageous in digital image processing in that no assumption is made about the statistical properties of the images, and they are thus widely applicable to a variety of dataset types. In addition, ANNs learn adaptively through examples and have a high tolerance to noisy or incomplete data (Jensen, 2005). ANN model development can proceed via either supervised or unsupervised means, and if adequate training data are available, supervised training may be readily performed. However, obtaining reliable training data in remote-sensing applications is often problematic (Congalton and Green, 1999), as a remote sensor image typically covers a large area, and only a limited number of training locations can be sampled in the field due to cost, time, personnel requirements, and various other logistical constraints, including potential restrictions on access to the study area. Unsupervised image-processing methods—including unsupervised ANNs—can be of significant utility in such circumstances (Filippi et al., 2009). Unsupervised ANNs are used in situations where the correct outputs may not be known, or if it is desired that the network discover or categorize regularities or features in the training data on its own. There is no teacher signal (Hassoun, 1995).

The unsupervised Kohonen self-organizing map (SOM) is a two-layer network, with an input fan-out layer, and an output layer (known as the Kohonen or competitive layer), and the method is based upon competitive learning. The Kohonen layer is comprised of a physical net of neurons located at fixed positions (i.e., intersections in a grid of square meshes). Adjacent neurons are assumed to have a Euclidean distance of unity. The input

layer has the same dimensionality as the pattern (feature) vector. The primary goal of the SOM is to capture the topology and model the probability distribution of the input vectors around the unit circle or hypersphere. The weight vectors act as a model for the probability distribution function. In an adaptive and topologically-ordered manner, the SOM converts patterns of arbitrary dimensionality (the pattern space) into the responses of one- or two-dimensional virtual arrays of self-organizing neurons (feature space), which are essentially probability distribution maps. This feature-mapping component reduces the dimensionality. It is a topology-preserving map, as it preserves the neighborhood relations of the input pattern. Input and output neurons are fully connected via the synaptic weights (Kohonen, 1988; Lin and Lee, 1996; Haykin, 2009).

Self-organization involves adaptively modifying the synaptic weights as a result of input excitations while abiding by a learning rule until a useful configuration is produced. In the output layer, or lattice, output neurons become selectively tuned to the presented input patterns during a competitive-learning procedure. The topologically-ordered output space entails neurons close to one another representing similar input patterns. Learning is accomplished by first choosing an output neuron that most closely matches the presented input pattern, then determining a neighborhood of excited neurons around the winner, and finally, updating all of the excited neurons (Haykin, 2009). Learning decreases with intercellular distance; the nearest neighbors to the primary classifying cell learn the pattern very well, while more distant cells learn the pattern less well (Hassoun, 1995).

Supervised learning may also be incorporated through a related technique referred to as Learning Vector Quantization (LVQ). Similar to SOM, all output nodes compete and the winning node is selected according to its similarity with the presented input pattern. Unlike SOM though, only the winning neuron is updated, and thus, the resulting output feature space is not topologically ordered (Kohonen, 2001). If training data are available, often the SOM analysis is performed first, and then LVQ is applied to fine-tune the feature map.

This chapter provides a brief overview of the applications of SOM-based techniques and related self-organizing methods used in remote sensing, and it demonstrates the use of SOMs in remote sensing through a case study involving the classification of a Landsat Enhanced Thematic Mapper Plus (ETM+) surface reflectance image.

2. Overview of SOM-based and SOM-related Applications in Remote Sensing

The basic unsupervised SOM algorithm has been applied in various remote-sensing studies (Table 1). For example, it has been used for identifying characteristics ocean current patterns over semidiurnal, diurnal and sub-tidal time scales (Liu et al., 2006) and for identifying synoptic-scale wind patterns and sea surface temperature patterns (Richardson et al., 2003). In another oceanographic remote-sensing study, Marques and Chen (2003) proposed a Kohonen SOM-based approach for detecting the borders of eddies in the Mediterranean. These Mediterranean Water Eddies strongly affect Atlantic Ocean hydrodynamics and are important in the transport of particles, live organisms, and suspended material. The objective was to discover image pixels on cluster boundaries, rather than the clusters themselves. Compared with a conventional Canny gradient edge-detector, the SOM detected the more significant and continuous borders. The Canny algorithm did not detect all borders at high threshold values and generated excessive noise at low threshold values. The SOM was thus more robust to noise and ill-defined/amorphous borders (Marques and

Chen, 2003). SOMs have also been used in synoptic climatology to locate archetypal points describing the multi-dimensional distribution function of gridded sea level pressure data. In Hewitson and Crane (2002), the SOM nodes represented important regional-scale circulatory features. Boekaerts et al. (1995) used an unsupervised, one-dimensional SOM in an autoadaptive mono-spectral cloud identification scheme employing Meteosat imagery. In terrestrial remote sensing, atmospheric features often constitute a source of noise, and minimizing such effects are typically desirable. For instance, monitoring vegetation cover and bare agricultural soils during the intercrop season is important for soil and water quality-management purposes, particularly in agriculture-intensive areas. Latif et al. (2008) identified and monitored bare soil interannual spatial variation at the regional scale using low-resolution satellite time-series data; however, such images are often adversely affected by clouds and shadows. The SOM was used to predict spectral values of pixels contaminated by weather conditions (i.e., clouds and cloud shadows) for posterior bare-soil mapping (Latif et al., 2008). In other terrestrial applications, the SOM algorithm has also been applied to detecting and characterizing yardangs, which are streamlined, elongated, wind-abraded ridges (Ehsani and Quiel, 2008), as well as to land-cover classification in an agricultural region and mapping fire-damaged forests (Furrer et al., 1994). Additionally, Kohonen's SOM has been used for automatically generating texture-based visual thesauri of massive archives remote-sensor images (e.g., aerial photographs and Advanced Very-High Resolution Radiometer (AVHRR) images) (Ramsey et al., 1999).

Authors	Year	Application
Ehsani and Quiel	2008	Detecting and characterizing yardangs
Latif et al.	2008	Predicting spectral values of data contaminated by weather conditions
Liu et al.	2006	Identifying ocean current patterns over semidiurnal, diurnal and sub-tidal time scales
Marques and Chen	2003	Detecting Mediterranean Water Eddy borders
Richardson et al.	2003	Identifying synoptic-scale wind patterns and sea surface temperature patterns
Hewitson and Crane	2002	Synoptic climatology
Ramsey et al.	1999	Creating texture-based visual thesauri of image collections
Boekaerts et al.	1995	Cloud identification
Furrer et al.	1994	Mapping fire-damaged forests; agricultural/land-cover classification

Table 1. Basic SOM used in various example remote-sensing application domains

Applications of SOM variants in remote sensing have also been demonstrated (Table 2), and such SOM variants have been prompted by SOM limitations. For example, although the SOM has been a stable ANN model in high-dimensional data analysis, its utility is limited

by its static network architecture and its limited ability to represent hierarchical relations in the data (Hong et al., 2006). Thus, a growing hierarchical SOM (GHSOM) has been proposed that employs a hierarchical structure comprised of multiple layers, where each layer is composed of one or more SOMs. Based on the input data distributions, the size of the sub-layers dynamically adapts during network learning, and feature maps are added as needed. This layered architecture explicitly represents hierarchical relations between input data (Hong et al., 2006). The GHSOM is advantageous for processing large remote-sensor data sets and increasing image classification accuracy (Liu et al., 2006). GHSOM has been applied to the problem of cloud classification in satellite images (Hong et al., 2006) and the analysis of sea surface temperature patterns (Liu et al., 2006). Growing SOM (GSOM) changes its structure during the learning process by adding neurons to the output space, and it was successfully applied to the problem of unsupervised classification of a multispectral Landsat Thematic Mapper (TM) image (acquired over Colorado, USA, with multiple vegetation and soil classes) and a hyperspectral Airborne Visible/Infrared Imaging Spectrometer (AVIRIS) scene (acquired over the Lunar Crater Volcanic Field (LCVF), Nevada, USA, which is dominated by mineralogic/geologic materials) (Vilmann et al., 2003). Another SOM variant involves magnification control through the use of an additional magnification control parameter, and the technique was applied to the same Landsat TM and AVIRIS scenes as GSOM (Vilmann et al., 2003). Another SOM variant, Multiple SOMs (MSOMs), fuse several small feature maps, each explicitly representing different cluster regions, thus better approximating each region, and thus better dealing with region borders (Wan and Fraser, 1999; Wan and Fraser, 2000). MSOMs vary in the type of SOM used and in the way the small feature maps are fused, which can be performed in accordance with unsupervised, supervised, or hybrid methods. MSOMs have been used for multisource data fusion and compound classification of a bitemporal, agricultural remote-sensor image data set (Wan and Fraser, 1999). In Wan and Fraser (2000), the applicability of MSOMs was demonstrated through land-cover classification of an AVIRIS image and a joint spatio-temporal classification of two Landsat TM images. A modified SOM based on the work of Hillermeier et al. (1994), which introduced self-organized lateral network connections, has also been used to perform land-cover classification, including mapping fire-damaged forests (Olbert et al., 1995; Schaale and Furrer, 1995).

Whereas the above studies demonstrate the applicability of SOM variants to remote-sensing image analysis, other studies specifically modify the basic/original SOM for achieving specialized tasks. For example, the Self-Organized Road Mapping (SORM) algorithm was developed for road extraction from high-resolution multispectral images where a pre-classified image is used as the input to a K-medians clustering algorithm, and the cluster locations are used as the input to a modified one-dimensional SOM (Doucette et al., 2008). Another SOM variant, the PicSOM, is used for the identification of specified anthropogenic structures such as buildings and for change detection using high-resolution images (Molinier et al., 2007). The technique involves the decomposition of images into subsets representing different structures and storing them in an image database. The stored image subsets are then used for SOM training, which subsequently allows for querying the image database for spatial structures of interest.

Supervised SOM is achieved by first coarse-tuning the feature map through unsupervised learning, and then fine-tuning the map using any supervised LVQ learning method (Kohonen, 1988). During LVQ training, the weights of the reference nodes are updated

Authors	Year	SOM Variant	Application
Doucette et al.	2008	SORM	Road extraction
Molinier et al.	2007	PicSOM	Detection of buildings; change detection
Hong et al.	2006	GHSOM	Cloud characterization
Liu et al.	2006	GHSOM	Sea surface temperature pattern analysis
Vilmann et al.	2003	GSOM Magnification control	Land-cover classification
Wan and Fraser	2000	MSOMs	Joint spatio-temporal agricultural classification
Wan and Fraser	1999	MSOMs	Multisource data fusion and joint spatio-temporal agricultural classification
Olbert et al.	1995	Population-coded, one-layer model of associative memory	Mapping fire-damaged forests
Schaale and Furrer	1995	Population-coded, one-layer model of associative memory	Mapping fire-damaged forests; land cover classification

Table 2. Example SOM variants applied to remote sensing

according to samples of specified classes. The originally-proposed LVQ1, LVQ2, and LVQ3 learning methods and their modifications have been applied to remote-sensing problems (Table 3). Fine-turning the classification of a Landsat TM image with LVQ performed better than a maximum likelihood classification (MLC) (Ji, 2000). Generalized relevance LVQ (GRLVQ) combines two important modifications of the Kohonen LVQs (Villmann et al., 2003). First, the cost function is minimized through stochastic gradient descent, and second, input weights are applied to obtain scaling of the input dimensions. In Villmann et al. (2003), GRLVQ was applied to Landsat TM and AVIRIS images for the purposes of land-cover/vegetation classification and geologic-mapping, respectively. The results were compared to the results obtained through GSOM and Magnification Control SOM (noted above), and it was demonstrated that GRLVQ produced the highest classification accuracy (Villmann et al., 2003). GRLVQ was further improved for use with high-dimensional hyperspectral images (Mendenhall and Merényi, 2008). GRLVQ Improved (GRLVQI) addresses the problem of poorly-utilized neurons by modifying the selection of winning neurons to encourage selection of infrequent winners. The technique was applied to a hyperspectral geologic AVIRIS image of the Lunar Crater Volcanic Field (LCVF) (noted above), and it was determined that higher classification accuracy can be obtained with GRLVQI relative to GRLVQ. The authors showed that relevance-selected features maintain or improve the performance of even basic classifiers (Mendenhall and Merényi, 2008).

Authors	Year	Technique	Application
Mendenhall and Merenyi	2008	GRLVQI	Land-cover classification
Vilmann et al.	2003	GRLVQ	Land-cover classification
Ji	2000	SOM/LVQ	Land-cover classification

Table 3. Supervised LVQ applications in remote sensing

The Kohonen self-organizing feature map, also referred to as the Kohonen clustering network (KCN) (Lin and Lee, 1996), exhibits some advantageous properties. For instance, weight vectors work to represent the natural tendency of a cluster of data points. It is unsupervised in that it requires no "teacher" during learning. KCN inherently reduces the dimensionality of the data by compressing and mapping it to the Kohonen layer, or mapping cortex. In addition, it always orients itself along the principal axes, but does not require the orthogonal and linear assumptions of standard principal component analysis (PCA). Thus, superior performance can be accrued when arbitrarily deformed distributions are present (Schaale and Furrer, 1995). And finally, KCN utilizes a parallel architecture, and it provides a graphical organization of topologically-preserved relationships. However, the KCN also entails several significant disadvantages: KCN is sequential, the termination strategy is artificial, and parameters such as the learning rate and the neighborhood size are selected heuristically, which does not ensure optimal performance. Convergence can be slow, and the network may not necessarily handle complexity accurately (Tsao et al., 1994).
Therefore, some attempts have been made in the ANN literature to address these disadvantages. For instance, Huntsberger and Ajjimarangsee (1990) first proposed a scheme that was essentially a partial integration of KCN and fuzzy *c*-means clustering (FCM), where the learning rates $\{\alpha_{ik}\}$ were replaced with fuzzy membership values $\{u_{ik,t}\}$ calculated with FCM. This approach was referred to as the Partial Fuzzy Kohonen Clustering Network (PFKCN). One extra layer, a membership layer, was added to the output layer, or *distance* layer, of the original KCN. The output u_{ik} represented the degree to which the input pattern x_k belonged to cluster I. During the learning process, however, in order to derive cluster centers that are the same as those of the FCM algorithm, it was necessary to feed these outputs back to update the cluster centers via a feedback path (Huntsberger and Ajjimarangsee, 1990). Note that m, the weighting exponent, still had to be chosen, and the approach lacked theoretical foundations and formal derivations (Bezdek and Pal, 1995). No properties pertaining to convergence were established; termination was forced. The fuzzy LVQ (FLVQ) algorithm (Bezdek and Pal, 1995) addresses these PFKCN shortcomings. Baraldi et al. (1998) gives a comparison of the properties of SOM and FLVQ algorithms.
Approaches based on the integration of ANN learning and fuzzy systems have been investigated in remote-sensing application domains (Table 4). Unsupervised fuzzy LVQ (FLVQ) combines FCM and unsupervised LVQ, and the technique has been applied to the task of coastal vegetation classification using hyperspectral AVIRIS surface reflectance image data (Filippi and Jensen, 2006). The effect of using continuum-removed AVIRIS image spectra as input to FLVQ for the same task was investigated in Filippi and Jensen (2007). Membership values are calculated for pixels according to a fuzzy *c*-partitioning of the

feature space, where, in a remote-sensing context, the feature vectors correspond to the set of input image bands (Filippi and Jensen, 2006). A modification of FLVQ incorporates Gaussian membership functions for calculating fuzzy membership grades for input pixels and consequently uses fuzzy membership grade and not Euclidean distance to estimate closeness of input pixels to output clusters (Qui and Jensen, 2004). Since Gaussian membership functions are used in fuzzifying Kohonen's LVQ2 algorithm, the Gaussian LVQ (GLVQ) employs supervised learning, though it can internally utilize unsupervised learning to generate natural clusters. Further modification of GLVQ uses training data to initialize the weights of the network and also ensures that only the winning neuron is updated (Qui, 2008). The improved GLVQ has been applied to Hyperion satellite image land-cover classification and achieved better classification accuracy compared to standard hyperspectral classification algorithms. Other FLVQ studies include Blonda et al. (1995) and Ceccarelli et al. (1995), where it was employed for Landsat image segmentation and SAR texture data classification; FLVQ was then known as the fuzzy Kohonen clustering network (FKCN). Baraldi et al. (1998) used FLVQ and SOM to classify 18 multitemporal Landsat TM bands into general land-cover categories. Benz (1999) proposed an adaptive system that integrated FLVQ with a supervised fuzzy distribution estimator to classify single- and multi-channel SAR data to identify flooded areas, water, and various land-cover classes.
FLVQ employs fuzziness in the learning rate and update strategies, but not typically in producing fuzzy outputs (Filippi and Jensen, 2006). Thus, remote-sensing studies utilizing FLVQ and related algorithms usually generate crisp, or hard, image classification results.

Authors	Year	Technique	Application
Qui	2008	GFLVQ Modified	Land-cover classification
Filippi and Jensen	2007	FLVQ	Coastal vegetation classification with continuum-removed spectra
Lee and Lathrop	2006	SOM-LVQ-GMM	Subpixel land cover
Filippi and Jensen	2006	FLVQ	Coastal vegetation classification
Qui and Jensen	2004	GFLVQ	Land-cover classification
Benz	1999	FLVQ integrated with supervised fuzzy distribution estimator	Flooded-area and land-cover classification
Baraldi et al.	1998	FLVQ and SOM	Land-cover classification
Ceccarelli et al.	1995	FKCN/FLVQ	Land-cover classification
Blonda et al.	1995	FKCN/FLVQ	Land-cover classification

Table 4. Fuzzy LVQ and related studies in remote sensing

Subpixel image analysis estimates the percentage of different land covers within a pixel. To estimate the percent cover of impervious surface, lawn, and woody tree cover in urban/suburban areas, Lee and Lathrop (2006) fine-tune a SOM with LVQ and then construct a Gaussian kernel density function for each SOM node. Using the Gaussian Mixture Model, the posterior probabilities of the land-cover classes were computed for each pixel. To apply the proposed approach to subpixel analysis, the authors equated the percent land-cover type composition with the posterior probability.

Other hybrid techniques have been used in remote sensing (Table 5). SOM, LVQ, fuzzy clustering, and parametric and non-parametric Bayesian approaches are used for deriving clusters, and selection of the best partitioning is accomplished through fuzzy multi-criteria decision-making (Guijarro and Pajares, 2009). The technique is applied to texture extraction from natural images. In Gonçalves et al. (2008), cluster analysis was performed on a set of SOM prototypes, rather than operating directly upon the original image patterns. Specifically, SOM was applied first; agglomerative hierarchical clustering with restricted connectivity was then applied to the SOM grid. Another hybrid approach uses wavelet fusion as a pre-processing step, and the results are used as input to SOM, which is then fine-tuned through LVQ (Bagan et al., 2008). Applying the technique for land classification of an ASTER image improved classification accuracy compared to using the original ASTER reflectance bands as input. A cascaded architecture of neural fuzzy neural networks with feature mapping (CNFM) involves feature extraction from multispectral images and inputting the features to an unsupervised SOM, which performs dimensionality reduction; the result is fed to a supervised neural fuzzy network which performs final clustering (Lin et al., 2000). Other attempts have also involved the Kohonen SOM being used for initial clustering as part of a modular network architecture, e.g., where supervised approaches have utilized a self-organizing component. Yoshida and Omatu (1994) used the SOM for initial clustering before the data was fed into a Multiple-Layer Feedforward Network (MLFN) trained with back-propagation (BP); in concert with geographical data, training areas were selected more precisely using the clustering results from the SOM. Blonda et al. (1996) used the Kohonen SOM as part of a modular neural classification system. A single-layer perceptron (SLP) (i.e., a multilayer perceptron (MLP) with no hidden neurons) was used as the supervised module. In another study, the weights at convergence of a Kohonen's self-organizing module have been used to initialize the weights between the input and hidden layers of a backpropagation ANN; this markedly increased the convergence rate, and thus decreased backpropagation network training time (Li and Si, 1992). Solaiman and Mouchot (1994) compared the MLC with the Kohonen SOM, LVQ2, MLP, and an unsupervised/supervised hybrid HLVQ network (consisting of an unsupervised SOM and a supervised LVQ2 network) in classifying an agricultural Landsat TM scene. In general, the hybrid method and the MLC produced the best results, but the SOM was comparable. These methods exceeded the classification accuracy of the MLP by about 10%. Ito and Omatu (1997) partitioned the training data to where a separate self-organizing competitive layer was assigned to each class. Once training was completed, a *k*-nearest neighbor approach was invoked such that *k* winner neurons were selected in order of minimum distance between the input vectors and the neuron weights. Although an artificial convergence criterion was used during self-organization, favorable results were still obtained. These modular systems have generally been successful when applied to multispectral remote-sensor images.

Authors	Year	Technique	Application
Guijarro and Pajares	2009	Fuzzy Multicriteria Decision Making Approach	Classifying textures in natural images
Gonçalves et al.	2008	SOM and agglomerative hierarchical clustering	Land-cover classification
Bagan et al.	2008	Wavelet Fusion and SOM/LVQ	Land-cover classification
Lin et al.	2000	CNFM	Land-cover classification
Ito and Omatu	1997	Self-organizing neural network and *k*-NN	Urban land-cover classification
Blonda et al.	1996	SOM and SLP	Land-cover classification
Solaiman and Mouchot	1994	Hybrid HLVQ (SOM and supervised LVQ2 network)	Agricultural classification
Yoshida and Omatu	1994	SOM and MLP	Land-cover classification
Li and Si	1992	Self-Organizing Backpropagation (SOBP)	Land-cover classification

Table 5. SOM hybrid studies in remote sensing

3. Unsupervised Classification of Landsat ETM+ Image Data Using SOM

An unsupervised snow-cover classification of a Landsat 7 ETM+ image demonstrates the application of SOM in remote sensing.

3.1 Data description

The Landsat ETM+ image used in the study was acquired on 27 March, 2002 and covers part of Wisconsin and Minnesota. The National Land Cover Database (NLCD) 2001 shows the predominant land covers as agricultural (cultivated crops, pasture/hay) and deciduous forest (Figure 1) (MRLC, 2009). The Mississippi River and several of its tributaries are notable spatial features in the scene. Snow cover is present in the northern half of the image and is easily distinguishable in cyan in a false-color composite of Landsat 7 ETM+ bands 5, 4 and 2 as R,G,B (Figure 2).

Landsat 7 ETM+ was launched in 1999 by National Aeronautics and Space Administration (NASA) and images the Earth once every 16 days in eight spectral bands (NASA, 2009). Bands 1-5 and band 7 have spatial resolution at nadir of 30 meters; band 6, which is a thermal infrared band, has a 60-meter resolution; and band 8, a panchromatic band, has 15-meter resolution. Bands 1-3 record electromagnetic energy in the visible portion of the spectrum, band 4 in the near-infrared, and bands 5 and 7 in the short-wave infrared. Bands 6 and 8 were not used in the study because of the difference in spatial resolution. The six bands used in the study (Table 7) were first scaled to radiance, then an atmospheric correction was applied using the MODTRAN4 radiation transfer code (Jensen, 2005), and the image was georegistered to an orthorectified Landsat ETM+ image.

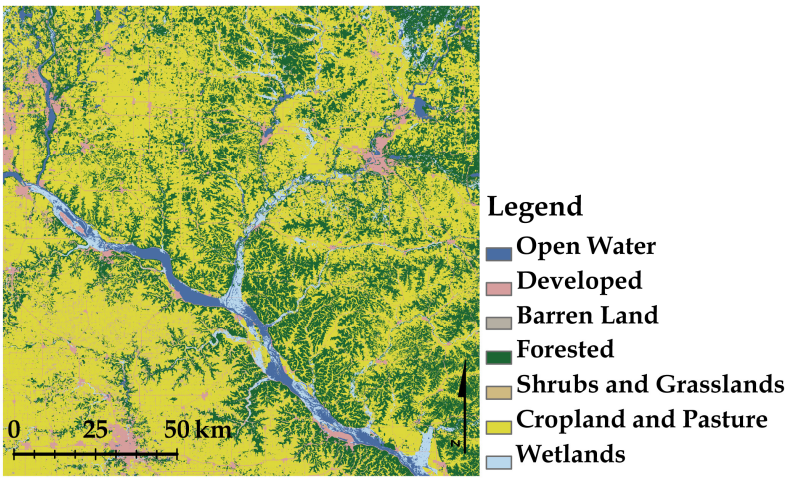


Fig. 1. Predominant land covers in the scene are cultivated crops, pasture/hay, and deciduous forests. The Mississippi River and some of its tributaries are major spatial features (Source: NLCD, 2001).

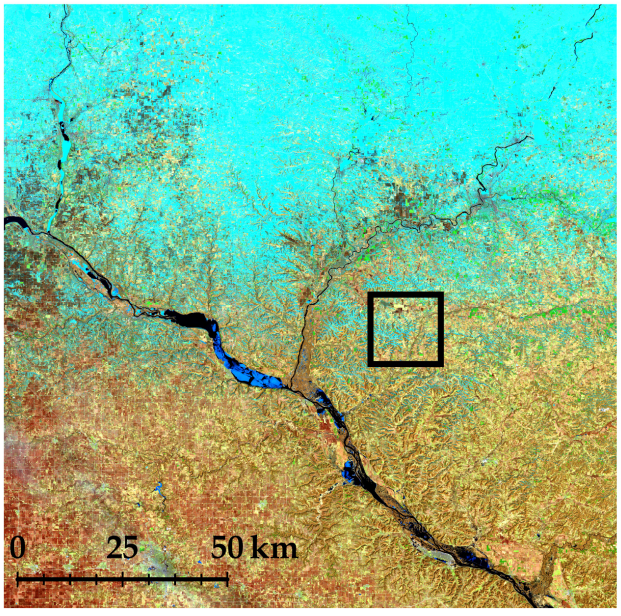


Fig. 2. A false-color composite image comprised of Landsat ETM+ bands 5, 4 and 2 as R,G,B shows snow cover as cyan colors in the northern half of the image. The square box represents the subset in Figure 4.

Input Data	Band pass (μm) / Equation
Landsat ETM+ Band 1	0.45 - 0.52
Landsat ETM+ Band 2	0.53 - 0.61
Landsat ETM+ Band 3	0.63 - 0.69
Landsat ETM+ Band 4	0.78 - 0.90
Landsat ETM+ Band 5	1.55 - 1.75
Landsat ETM+ Band 7	2.09 - 2.35
NDSI	$\frac{Band\ 4 - Band\ 6}{Band\ 4 + Band\ 6}$
NDVI	$\frac{Band\ 4 - Band\ 3}{Band\ 4 + Band\ 3}$

Table 7. Two input-combination permuations to SOM were assessed. The first one included six of the Landsat 7 ETM+ bands (bands 1-5 and band 7), and the second data set included two additional normalized difference index images, which have been demonstrated to be useful in mapping snow in forests (Klein et al., 1998).

3.2 Methodology

Image analysis was performed in the IDRISI 16 GIS/remote-sensing software package using the implemented SOM module, which is modeled after the method described in Kohonen (1990). The neural network was run with an initial neighborhood radius of 17.97; a minimum learning rate of 0.5; a maximum learning rate of 1; and a 12 × 12 lattice of output neurons. A k-means clustering was applied to the SOM results, and a maximum of 10 k-means classes/clusters was specified.

Two networks were developed with the same parameters, but with different input combinations. For the first input combination, inputs were the six Landsat ETM+ surface reflectance bands (Table 7). For the SOM with this input combination, the quantization error, which measures the average distance between each input and the neuron to which it is mapped, was 0.0775. For the second input combination, two additional normalized difference index bands were added (Table 7), lowering the quantization error to 0.0169. Reflectance bands 3 and 4 are used to calculate the Normalized Difference Vegetation Index (NDVI) which captures the inverse relationship between reflectance in the red and near-infrared portions of the electromagnetic spectrum associated with healthy green vegetation (Jensen, 2005). The index is useful in distinguishing vegetated areas that have higher NDVI values from non-vegetated or sparsely-vegetated areas with lower NVDI values. Normalized Difference Snow Index (NDSI) is calculated using bands 4 and 6, and, similarly to NDVI, it uses the inverse relationship between snow reflectance in the visible and near-infrared (Hall, 1995). Adding the two index images does not increase the information

content of the inputs, but it does increase the redundancy in the input data. Redundancy and structure in the input data is one of the principles of self-organization (Haykin, 2009), and therefore, the second input combination would be expected to produce better results.
The performance of SOM in detecting snow-covered pixels using the two input combinations was compared to the MODIS snow-mapping algorithm (SNOMAP) which has a long history of use in mapping snow and is the standard algorithm for producing global daily snow maps from MODIS (Hall, 1995; Hall et al., 2002). SNOMAP maps snow primarily using the NDSI, but a combination of NDSI and NDVI improves snow mapping in forests (Klein et al., 1998). Also, SNOMAP excludes water pixels from analysis (Hall, 1995). To facilitate the comparison of SOM results to SNOMAP, water is masked in the current study.

3.3 Results and discussion

The input combination including only reflectance bands returned 10 classes that were subsequently combined into snow-covered and snow-free areas. The resulting snow map has a 92% overall agreement with SNOMAP (Table 8). Commission errors, where SOM maps snow missed by SNOMAP, were 24.56%, whereas omission errors, pixels mapped as snow by SNOMAP but missed by SOM, is much lower (only 3.71%).
The second input combination which includes the additional index bands returned only three classes. Two of the classes are easily identified as snow and snow-free. The third class represented a combination of water and snow-covered pixels. Excluding water from the analysis, the remaining third class pixels were considered snow. (Figure 3). The overall agreement between the second SOM result and SNOMAP is 88% (Table 9). No omission errors occurred, meaning SOM did not miss any of the snow-covered areas mapped by SNOMAP. However, the commission errors were high, e.g., 35.38% for snow-covered, which is larger than the same error for the reflectance-only input combination.
Both input combinations resulted in a larger area mapped as snow compared to SNOMAP, with the input combination with additional index bands mapping the largest snow-covered area. SNOMAP mapped 4,499 km^2 as snow; SOM with the first input data combination mapped 5,703 km^2; and SOM with the second input combination mapped 6,915 km^2 as snow.
The errors of commission between the input combination with additional index bands and SNOMAP were examined visually (Figure 4). The area mapped as snow by SOM appears to be snow-covered in the false color-composite of Landsat bands 5, 4 and 2, but failed to be mapped as snow by SNOMAP. So even though the second input combination has lower

Reflectance Bands Only	**SNOMAP**	
	Snow	*Snow-Free*
Snow	4,779,553 (21.075%)	1,556,011 (6.86%)
Snow-Free	184,057 (0.81%)	16,167,435 (71.26%)
Overall agreement: 92%		

Table 8. Confusion matrix between SNOMAP and SOM with reflectance bands only as input shows number of pixels that are in agreement or disagreement. Numbers in parenthesis indicate percentage of total count.

Reflectance and Index Bands	SNOMAP	
	Snow	*Snow-Free*
Snow	4,963,610 (21.88%)	2,718,029 (11.98%)
Snow-Free	0 (0%)	15,005,417 (66.14%)
Overall agreement: 88%		

Table 9. Confusion matrix between SNOMAP and SOM with both reflectance bands and index bands as input, indicating number of pixels and percent agreement and disagreement.

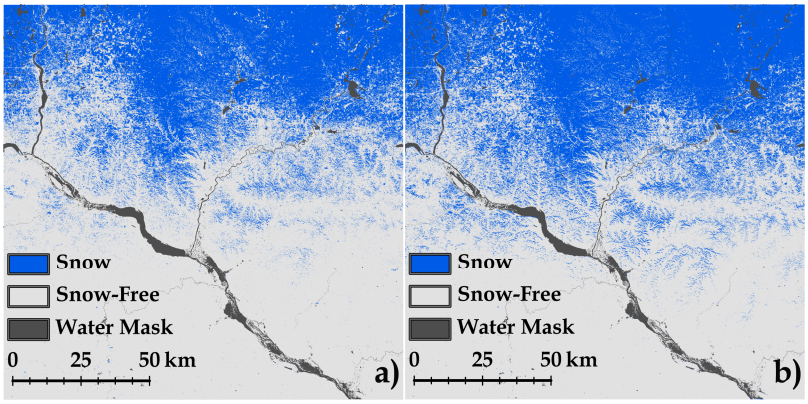


Fig. 3. Snow maps produced by SNOMAP (a) map lesser snow-cover extent compared to the snow maps produced by SOM with both reflectance bands (not illustrated) and index bands (b) as input.

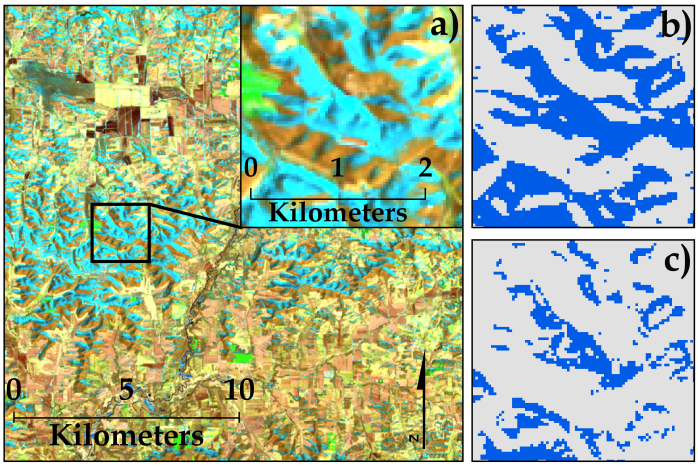


Fig. 4. Portion of false-color image composite examined (a). Comparing the results of SOM with both reflectance bands and index bands as input (b) and SNOMAP (c) shows that the larger snow-cover extent derived by SOM appears more correct.

overall accuracy as compared to SNOMAP than the first input combination, it appears to provide more accurate snow mapping. However, further validation of the SOM results requires the use of field observations or higher-resolution images. Note that the "errors" stated in study may not technically be considered as errors per se, as we treated the SNOMAP result as reference data, rather than obtaining ground reference data.

4. Conclusion

SOM-based and SOM-related techniques have been applied in remote sensing. Image-analysis tasks range from identifying synoptic-scale ocean or atmospheric characteristics to land-cover classification. The utility of the Kohonen SOM as an unsupervised classification technique was demonstrated here by generating a snow map based upon a Landsat 7 ETM+ image. SOM-mapping of snow compares favorably to the widely-used SNOMAP algorithm, which has a considerable heritage in mapping snow from satellite images. In the single test image, the SOM mapped snow that SNOMAP missed. Better results were achieved when the input bands were complimented by index bands, which increased the redundancy in the input.

5. References

Bagan, H.; Wang, Q.; Watanabe, M.; Karneyarna, S. & Bao, Y. (2008). Land-cover classification using ASTER multi-band combinations based on wavelet fusion and SOM neural network. *Photogrammetric Engineering and Remote Sensing,* Vol. 74, No. 3, 333-342, ISSN 0099-1112

Baraldi, A.; Blonda, P.; Parmiggiani, F.; Pasquariello, G. & Satalino, G. (1998). Model transitions in descending FLVQ. *IEEE Transactions on Neural Networks*, Vol. 9, No. 5, 724– 738, ISSN 1045-9227

Benz, U. C. (1999). Supervised fuzzy analysis of single- and multichannel SAR data. *IEEE Transactions on Geoscience and Remote Sensing*, Vol. 37, No. 2, 1023–1037, ISSN 0196-2892

Bezdek, J. C. & Pal, N. R. (1995). Two soft relatives of learning vector quantization. *Neural Networks*, Vol. 8, No. 5, 729–743, ISSN 0893-6080

Blonda, P.; Bennardo, A.; Satalino, G. & Forgia, V. (1995). Application of the unsupervised fuzzy Kohonen clustering network for remote sensed data segmentation. In A. Bonarini; D. Mancini; F. Masulli & A. Petrosino (Eds.), Proceedings of WILF '95, Italian workshop on fuzzy logic: New trends in fuzzy logic, pp. 143– 150, Naples, Italy, September 1995, World Scientific, River Edge, NJ

Blonda, P.; la Forgia, V.; Pasquariello, G. & Satalino, G. (1996). Feature extraction and pattern classification of remote sensing data by a modular neural system. *Optical Engineering*, Vol. 35, No. 2, 536-542, ISSN 0091-3286

Boekaerts, P.; Nyssen, E. & Cornelis, J. (1995). Autoadaptive mono-spectral cloud identification in Meteosat satelliete images, In *Image and Signal Processing for Remote Sensing II*, J. Desachy, Ed., Proceedings of SPIE, pp. 259-271, Vol. 2579, Paris, France, September 1995

Ceccarelli, M.; Farina, A. & Petrosino, A. (1995). Fuzzy unsupervised terrain classification based on a multiresolution approach. In A. Bonarini, D. Mancini, F. Masulli & A. Petrosino (Eds.), Proceedings of WILF'95, Italian workshop on fuzzy logic: New trends in fuzzy logic, pp. 151– 159, Naples, Italy, September 1995, World Scientific, River Edge, NJ

Congalton, R.G. & Green, K. (1999). *Assessing the Accuracy of Remotely Sensed Data: Principles and Practices*, Lewis, Boca Raton, FL, ISBN 0-873-71986-7

Doucette, P.; Agouris, P. & Stefanidis, A. (2008). Self-Organising Map Principles Applied Towards Automatic Road Extraction from Remotely Sensed Imagery, In*Self-Organising Maps. Applications in Geographic Information Science,* Skupin, A., (Ed.), 177-194, Wiley, ISBN 978-0-470-02167-5, West Sussex

Ehsani, A. H. & Quiel, F. (2008). Application of self organizing map and SRTM data to characterize yardangs in the Lut desert, Iran. *Remote Sensing of Environment,* Vol. 112, No. 7, 3284-3294, ISSN 0034-4257

Filippi, A. M.; Brannstrom, C.; Dobreva, I.; Cairns, D. M. & Kim, D. (2009). Unsupervised fuzzy ARTMAP classification of hyperspectral Hyperion data for savanna and agriculture discrimination in the Brazilian Cerrado. *GIScience & Remote Sensing*, Vol. 46, No. 1, 1-23, ISBN 1548-1603

Filippi, A. M. & Jensen, J. R. (2006). Fuzzy learning vector quantization for hyperspectral coastal vegetation classification. *Remote Sensing of Environment,* Vol. 100, No. 4, 512-530, ISSN 0034-4257

Filippi, A. M. & Jensen, J. R. (2007). Effect of continuum removal on hyperspectral coastal vegetation classification using a fuzzy learning vector quantizer. *IEEE Transactions on Geoscience and Remote Sensing*, Vol. 45, No. 6, 1857-1869, ISSN 0196-2892

Furrer, R.; Barsch, A.; Olbert, C. & Schaale, M. (1994). Multispectral imaging of land surface. *GeoJournal*, Vol. 32, No. 1, 7-16, ISSN 0343-2521

Gonçalves, M. L.; Netto, M. L. A.; Costa, J. A. F. & Zullo Júnior, J. (2008). An unsupervised method of classifying remotely sensed images using Kohonen self-organizing maps and agglomerative hierarchical clustering methods. *International Journal of Remote Sensing,* Vol. 29, No. 11, 3171-3207, ISSN 0143-1161

Guijarro, M. & Pajares, G. (2009). On combining classifiers through a fuzzy multicriteria decision making approach: Applied to natural textured images. *Expert Systems with Applications,* Vol. 36, No. 3, 7262-7269, ISSN 0957-4174

Hall, D. K.; Riggs, G. A. & Salomonson, V. V. (1995). Development of methods for mapping global snow cover using moderate resolution imaging spectroradiometer data. *Remote Sensing of Environment,* Vol. 54, No. 2 , 127-140, ISSN 0034-4257

Hall, D. K.; Riggs, G. A.; Salomonson, V. V.; DiGirolamo, N. E. & Bayr, K. J. (2002). MODIS snow-cover products. *Remote Sensing of Environment,* Vol. 83, No. 1-2, 181-194, ISSN 0034-4257

Hassoun, M. H. (1995). *Fundamentals of Artificial Neural Networks.* MIT Press, Cambridge, MA, ISBN 0-262-08239-X

Haykin, S. (2009). *Neural Networks and Learning Machines,* Pearson Education, ISBN 0-13147139-2, Upper Saddle River

Hewitson, B. C. & Crane, R. G. (2002). Self-organizing maps: applications to synoptic climatology. *Climate Research,* Vol. 22, No. 1, 13-26, ISSN 0936-577X

Hillermeier, C.; Kunstmann, N.; Rabus, B. & Tavan, P. (1994). Topological feature maps with self-organized lateral connections: a population-coded one-layer model of associative memory. *Biological Cybernetics*, Vol. 72, 103-117, No. 2, 103-117, ISSN 0340-1200

Hong, Y.; Chiang, Y. M.; Liu, Y.; Hsu, K. L. & Sorooshian, S. (2006). Satellite-based precipitation estimation using watershed segmentation and growing hierarchical self-organizing map. *International Journal of Remote Sensing,* Vol. 27, No. 23-24, 5165-5184, ISSN 0143-1161

Huntsberger, T. L. & Ajjimarangsee, P. (1990). Parallel self-organizing feature maps for unsupervised pattern recognition. *International Journal of General Systems*, Vol. 16, No. 4, 357-372, ISSN 0308-1079

Ito, Y. & Omatu, S. (1997). Category classification method using a self-organizing neural network. *International Journal of Remote Sensing,* Vol. 18, No. 4, 829-845, ISSN 0143-1161

Jensen, J. R. (2005). *Introductory Digital Image Processing: A Remote Sensing Perspective,* 3rd Ed., Pearson Education, ISBN 0-13-145361-0, Upper Saddle River

Ji, C. Y. (2000). Land-use classification of remotely sensed data using Kohonen Self-Organizing Feature Map neural networks. *Photogrammetric Engineering and Remote Sensing,* Vol. 66, No. 12, 1451-1460, ISSN 0099-1112

Klein, A. G.; Hall, D. K. & Riggs, G. A. (1998). Improving snow cover mapping in forests through the use of a canopy reflectance model. *Hydrological Processes,* Vol. 12, No. 10-11, 1723-1744, ISSN 0885-6087

Kohonen, T. (1988). *Self-Organization and Associative Memory,* Springer-Verlag, ISBN 0-387-18314-0, New York, Berlin, Heidelberg

Kohonen, T. (1990). The Self-Organizing Map. *Proceedings of the IEEE*, Vol. 78, pp. 1464-1480, ISSN 0018-9219, September 1990

Kohonen, T. (2001). *Self-Organizing Maps.* Springer-Verlag, ISBN 3-540-67921-9, New York, Berlin, Heidelberg

Latif, B. A.; Lecerf, R.; Mercier, G. & Hubert-Moy, L. (2008). Preprocessing of low-resolution time series contaminated by clouds and shadows. *IEEE Transactions on Geoscience and Remote Sensing,* Vol. 46, No. 7, 2083-2096, ISSN 0196-2892

Lee, S. & Lathrop, R. G. (2006). Subpixel analysis of Landsat ETM+ using Self-Organizing Map (SOM) neural networks for urban land cover characterization. *IEEE Transactions on Geoscience and Remote Sensing,* Vol. 44, No. 6, 1642-1654, ISSN 0196-2892

Li, R. & Si, H. (1992). Multi-spectral image classification using improved backpropagation neural networks, In *Proceedings of the IEEE International Geosciences and Remote Sensing Symposium*, (IGARSS '92), Vol. 2, pp. 1078-1080, Houston, TX, May 1992

Lin, C. T. & Lee, C. S. G. (1996). *Neural fuzzy systems: A neuro-fuzzy synergism to intelligent systems.* Prentice Hall P T R, Upper Saddle River, NJ, ISBN 0-132-35169-2

Lin, C. T.; Lee, Y. C. & Pu, H. C. (2000). Satellite sensor image classification using cascaded architecture of neural fuzzy network. *IEEE Transactions on Geoscience and Remote Sensing,* Vol. 38, No. 2, 1033-1043, ISSN 0196-2892

Liu, Y., Weisberg, R. H. & He, R. (2006). Sea surface temperature patterns on the West Florida Shelf using the growing hierarchical self-organizing maps. *Journal of Atmospheric and Oceanic Technology,* Vol. No. 2, 23, 325-328, ISSN 0739-0572

Liu, Y. G., Weisberg, R. H. & Shay, L. K. (2007). Current patterns on the West Florida Shelf from joint self-organizing map analyses of HF radar and ADCP data. *Journal of Atmospheric and Oceanic Technology,* Vol. 24, No. 4, 702-712, ISSN 0739-0572

Marques, N. C. & Chen, N. (2003) Border detection on remote sensing satellite data using self-organizing maps, *Proceedings of EPIA 2003 – 11th Portuguese conference on Artificial Intelligence*, pp. 294-307, ISBN 3-540-20589-6, Beja, Portugal, December 2003, Springer-Verlag, Berlin

Mendenhall, M. J. & Merényi, E. (2008). Relevance-based feature extraction for hyperspectral images. *IEEE Transactions on Neural Networks,* Vol. 19, No. 4, 658-672, ISSN 1045-9227

Molinier, M.; Laaksonen, J. & Hame, T. (2007). Detecting man-made structures and changes in satellite imagery with a content-based information retrieval system built on self-organizing maps. *IEEE Transactions on Geoscience and Remote Sensing,* Vol. 45, No. 4, 861-874, ISSN 0196-2892

MRLC, (2009). National Land Cover Database, http://www.mrlc.gov/index.php

NASA, (2009). Landsat 7: Science Data Users Handbook, http://landsathandbook.gsfc.nasa.gov/handbook/handbook_toc.html

Olbert, C.; Schaale, M. & Furrer, R. (1995). Mapping of forest fire damages using imaging spectroscopy. *Advances in Space Research*, Vol. 15, No. 11, (11)115-(11)122, ISSN 0273-1177

Qiu, F. (2008). Neuro-fuzzy based analysis of hyperspectral imagery. *Photogrammetric Engineering and Remote Sensing,* Vol. 74, No. 10, 1235-1247, ISSN 0099-1112

Qiu, F. & Jensen, J. R. (2004). Opening the black box of neural networks for remote sensing image classification. *International Journal of Remote Sensing,* Vol. 25, No. 9, 1749-1768, ISSN 0143-1161

Ramsey, M. C.; Chen, H.; Zhu, B. & Schatz, B. R. (1999). A collection of visual thesauri for browsing large collections of geographic images. *Journal of the American Society for Information Science,* Vol. 50, No. 9, 826-834, ISSN 1097-4571

Richardson, A. J.; Risien, C. & Shillington, F. A. (2003). Using self-organizing maps to identify patterns in satellite imagery. *Progress in Oceanography,* Vol. 59, No. 2-3, 223-239, ISSN 0079-6611

Schaale, M., & Furrer, R. (1995). Land surface classification by neural networks. *International Journal of Remote Sensing*, Vol. 16, No. 16, 3003-3031, ISSN 0143-1161.

Solaiman, B. & Mouchot, M. C. (1994). A comparative study of conventional and neural network classification of multispectral data, In *Proceedings of the International Geoscience and Remote Sensing Symposium* (IGARSS '94), Surface and Atmospheric Remote Sensing: Technologies, Data Analysis and Interpretation, pp. 1413-1415, ISBN 0-7803-1497-2, Pasadena, CA, August 1994

Tsao, E. C. -K.; Bezdek, J. C. & Pal, N. R. (1994). Fuzzy Kohonen clustering networks. *Pattern Recognition*, Vol. 27, No. 5, 757– 764, ISSN 0031-3203

Villmann, T.; Merenyi, E. & Hammer, B. (2003). Neural maps in remote sensing image analysis. *Neural Networks,* Vol. 16, No. 3-4, 389-403, ISSN 0893-6080

Wan, W. & Fraser, D. (1999). Multisource data fusion with Multiple Self-Organizing Maps. *IEEE Transactions on Geoscience and Remote Sensing,* Vol. 37, No. 3, 1344-1349, ISSN 0196-2892

Wan, W. J. & Fraser, D. (2000). A Multiple Self-Organizing Map scheme for remote sensing classification, *Proceedings of* Multiple Classifier Systems, First International Workshop, MCS 2000, pp. 300-309, ISBN 3-540-67704-6, Cagliari, Italy, June, 2000, Springer-Verlag, Berlin

Yoshida, T. & Omatu, S. (1994). Neural network approach to land cover mapping. *IEEE Transactions on Geoscience and Remote Sensing,* Vol. 32, No. 5, 1103-1109, ISSN 0196-2892

15

Segmentation of satellite images using Self-Organizing Maps

Mohamad Awad
National Council for Scientific Research-Center for Remote Sensing
Lebanon

1. Introduction

Remote sensing plays a key role in many domains devoted to observation of the Earth, such as land cover/use, agriculture monitoring, military battles, oceanography…etc. There are many types of acquisition systems which have different spatial, spectral, and temporal characteristics. Some of them are passive, such as Landsat, Spot, Ikonos, Quickbird, Orbview…etc. Others are active such as SAR (Space Airborne Radar). These systems have opened the field of applications since early 1970.

Image segmentation is the process of image division into regions with similar attributes (Pratt, 1991). It is an important step in image analysis chain with applications to pattern recognition, object detection, etc. Until recently, most of the segmentation methods and approaches are supervised such as Maximum A Posteriori (MAP) (Lopes et al., 1990) or Maximum Likelihood (ML) (Benediktson et al., 1990) with an average efficiency rate of about 85% (Perkins et al., 2000), (Zhang et al., 2003). In the supervised methods *a priori* knowledge is needed to get a successful segmentation process and sometime the required information may not be available. In addition, there are unsupervised methods which require many parameters and they are sensitive to noise such as Iterative Self-Organizing Map Data (ISODATA) (Tou & Gonzalez, 1974), and SEM (Mason & Pieczynski, 1993). In order to overcome the deficiencies found in many previously listed methods, Kohonen's Self-Organizing Maps (SOM) (Kohenen, 2001) is used to segment different satellites images. SOM is an unsupervised non-parametric Artificial Neural Network (ANN) method. The main characteristic of SOM is the ability to convert patterns of arbitrary dimensionality into the responses of two dimensional arrays of neurons. Another important characteristic of the SOM is that the feature map preserves neighborhood relations of the input pattern. Although the use of SOM in image segmentation is well reported in the literature, such as segmentation of printed fabric images (Xu & lin, 2002), or in sonar images (Yao et al.,2000), their application in satellite image segmentation is not widely known. One can cite the work of (Aria et al., 2004) which was used in the segmentation of Indian Remote Sensing "IRS" satellite image. The cooperative segmentation approach between K-means and SOM (Zhou et al., 2007) is a recent work where the role of K-means is to segment the image in the coarser scale, and then SOM will re-segment the image in the fine scale. In This method K-

means requires the pre-determination of cluster numbers and this kind of work can be taken care by SOM itself.
Another promising work is the cooperation between SOM and the Hybrid Genetic Algorithm (GA) (Awad et al., 2007). SOM provides many cluster centers and GA finds the optimal number of these centers. This cooperation was tested on different satellite images such as Landsat, Spot, Ikonos and the results were of high accuracy. However, the speed of image processing remains an important issue.
In this chapter, SOM based on a threshold technique and SOM cooperated with another method are tested on two different types of satellite images with different resolution. In the first method SOM is combined with the threshold based technique. In the cooperative segmentation method, SOM is combined with the Hybrid Dynamic Genetic Algorithm (HDGA) (Awad et al. 2009a) to segment the images. The goal is to check the efficiency of SOM with and without cooperation with another algorithm. In addition, the use of HDGA is to increase the speed of processing.
After this introduction, SOM complete details will be covered in the second section. The third section covers the combination of SOM and threshold based segmentation technique. The forth section covers the combination of SOM and the Hybrid Dynamic Genetic Algorithm. In the fifth section the experimental results are presented, and finally the conclusion.

2. Image segmentation using Self-Organizing Maps (SOM)

Kohonen's Self-Organizing Maps (SOM) (Kohenen, 2001) is an unsupervised neural network method. SOM converts patterns of arbitrary dimensionality into the responses of two dimensional arrays of neurons. One of the important characteristics of SOM is that the feature map preserves neighborhood relations of the input pattern. A typical SOM structure is given in Figure 1. It consists of two layers: an input layer and an output layer. The number of input neurons is equal to the dimensions of the input data. The output neurons are, however, arranged in a two-dimensional array.

Colors are one of the most important features considered in biological visual systems, since it is used to separate objects and patterns, even in conditions of equi-luminance (Levin, 1985). SOM is used to map patterns in a three-dimensional color (multi-bands) space to a two-dimensional space. In SOM, the input signals are n-tuples and there is a set of m cluster units (automatic empirical determination with respect to the size of the satellite image). Each input is fully connected to all units. The initial weights are random and small, and their contribution for the final state decreases with the decrease of the number of samples (Yin & Allinson, 1995). The network is composed of an orthogonal grid of cluster units (neurons), each is associated with three internal weights for the three layers of the satellite image. At each step in the training phase, the cluster unit with weights that best match the input pattern is elected as the winner usually by using minimum Euclidean distance as in Equation (1)

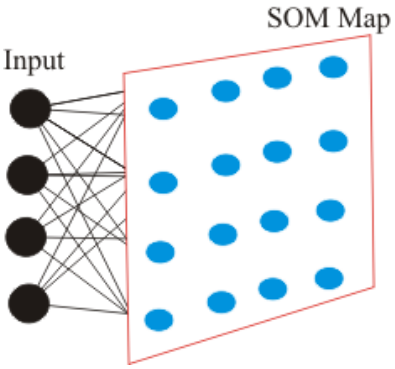


Fig. 1. Self-Organizing Map

$$\left\|\mathbf{x} - W_l^{\lfloor k \rfloor}\right\| = \min_i \left\|\mathbf{x} - W_i^{\lfloor k \rfloor}\right\| \tag{1}$$

Where $\mathbf{x}$ is the input vector, $W_l^{\lfloor k \rfloor}$ is the weight of the wining unit l at iteration k, and $W_i^{\lfloor k \rfloor}$ is the weight for neuron i at iteration k. The winning unit and a neighborhood around it are then updated in such a way that their internal weights be closer to the presented input. All the neurons within a certain neighborhood around the *leader* participate in the weight-update process. This learning process can be described by the iterative procedure as in Equation (2).

$$W_i^{\lfloor k+1 \rfloor} = W_i^{\lfloor k \rfloor} + H_{li}^{\lfloor k \rfloor}(\mathbf{x} - W_i^{\lfloor k \rfloor}) \tag{2}$$

Where $H_{li}^{\lfloor k \rfloor}$ is a smoothing kernel defined over winning neuron. This kernel can be written in terms of the Gaussian function as in Equation 3.

$$H_{li}^{\lfloor k \rfloor} = \alpha^{\lfloor k \rfloor} \exp\left(-\frac{d^2(l,i)}{2(\sigma^{\lfloor k \rfloor})^2}\right) \tag{3}$$

$H_{li}^{\lfloor k \rfloor} \to 0$ when $k \to To$, where To is the total number of iterations. $\alpha^{\lfloor 0 \rfloor}$ is the initial learning rate and it is equal to 0.1. The learning rate is updated every iteration. $\sigma^{\lfloor k \rfloor}$ is the search distance at iteration k, initially can be half the length of the network or the maximum of either the width or length of the image divided by two. $d(l,i)$ is the distance between the leader neuron l and its neighbor i. As learning proceeds, the size of the neighborhood should be diminished until it encompasses only a single unit.

After SOM neural network converges to a balance state, the original image is mapped from a high color space to a smaller color space. The number of colors in this space is equal to the number of neurons of SOM network. The final weights vectors in the map as the new sample space. This new data set is used for clustering, and allows determining a set of cluster centers.

3. SOM and the Threshold technique

In order to eliminate small clusters (clusters with few pixels) and to reduce over segmentation problem the following technique (T-Cluster) is implemented. This technique consists of several steps as follow:
1-After obtaining cluster centers by SOM the process of clustering starts by calculating the distance between the values of the cluster centers representing the sum of the three bands.
2-Two clusters are combined if the distance between their centers is less than a predefined threshold *T*.
3- If step two is correct, the minimum number of pixels is considered in the combination procedure, where the cluster with smaller number of pixels is merged with the larger one. Figure 2 shows more details of the merging procedure and Equation 4 explains distance calculation.

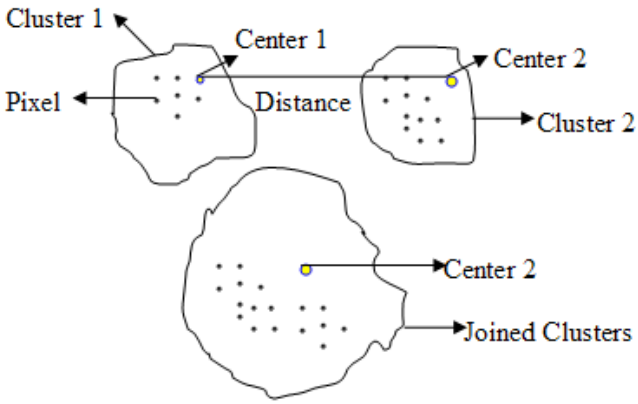


Fig. 2. Merging process according to the distance between cluster centers

$$d(V(P_i),V(P_j)) <= T \tag{4}$$

Where T is a predefined threshold and V (P_i) is the value of the three bands of the cluster center P_i. The value represents the sum of the resultant 3 weights obtained from running SOM each weight is multiplied by 255. V (P_j) is the value of the three bands of another cluster center P_j. These two cluster centers are combined together if the distance value is less than a predefined threshold T. The value of the final cluster is the cluster with higher number of pixels.
SOM and the threshold technique (T-Cluster) work sequentially in order to complete the segmentation process. In other words, working separately cannot complete the job correctly (Figure 3). SOM uses satellite image features to organize pixels in group. The highest peaks of the histogram are used as cluster centers and are provided to T-Cluster to deliver the final solution in the image segmentation process.
This method starts by reading a satellite image than it is provided to SOM to organize pixels in groups. The organized pixels are used by T-Cluster to obtain the final number of cluster centers (no under or over segmentation). TSOM fixes the problem of under and over segmentation which are caused by using SOM separately.

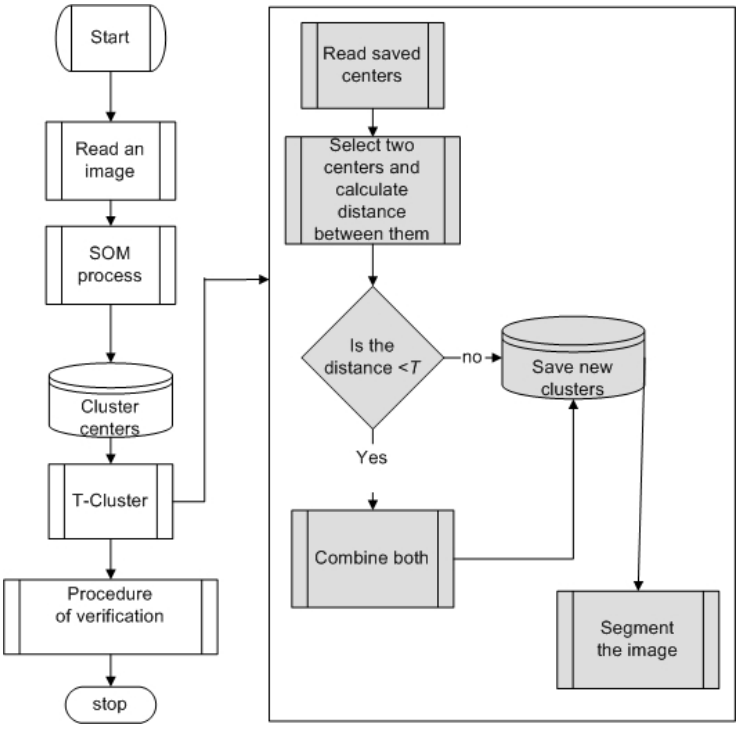


Fig. 3. SOM and T-Cluster sequential process

4. SOM and Hybrid Dynamic Genetic Algorithm

Accuracy obtained using only SOM in image segmentation may often be unsatisfactory (Awad et al., 2009b). So, in order to improve the result of satellite image segmentation, SOM and HDGA (Awad et al., 2009a) work sequentially in order to achieve the highest accuracy (See Figure 4 for complete details).

First, the process starts by reading a satellite image, and then SOM uses multi-component features of the image to organize the image pixels in groups. Each group value is used as a cluster center and is provided to the Hybrid Dynamic Genetic Algorithm (HDGA) for selecting the optimal solution to select the optimal image segmentation solution. HDGA is a modified Genetic Algorithm (GA) to become Hybrid by adding Hill-climbing process. Moreover, a heuristic process which increases the number of feasible solutions is added in order to speed up the process of escaping local optima solution. The chromosomes which form the population of the Hybrid Dynamic Genetic Algorithm (HDGA) consist of different number of cluster centers, which means that many different solutions are available. In the previous method the success of the segmentation process depends on the correct selection of two criteria:

1- The minimum number of pixels in each group and 2- The degree of similarity of the grey level values of the cluster centers.

These two criteria were necessary and they had to be adjusted in the previous method using the threshold technique. But, this concern is eliminated by using HDGA.
HDGA creates population of chromosomes where each four genes represent the cluster center provided by SOM and the other three genes represent the grey level value for each pixel in the three bands in the satellite image. In each iteration, the chromosomes are evaluated using (5) and the best solution is selected.

$$OF = Min(\sum_{j=1}^{k}\sum_{i=0}^{n}[\sum_{y=1}^{3}V(P_{jy}) - \sum_{z=1}^{3}V(px_{iz})]) \quad (5)$$

Where k is the number of the centers in a chromosome and $V(P_{jy})$ is the value of the cluster center P_j in each band of the 3 bands of the image. $V(px_{iz})$ is the grey level values of the pixel in each band of the 3 bands of the image to the left of the cluster center P_j in the chromosome. Variable n is the number of pixels in each cluster (…px_{i1} px_{i2} px_{i3} P_{j1} P_{j2} P_{j3} $px_{(i+1)1}$ $px_{(i+1)2}$ $px_{(i+1)3}$$P_{j1}$ P_{j2} P_{j3}…).
SOM-HDGA fixes the problem of under and over segmentation caused by using one method alone as will be proved later in the experiments.

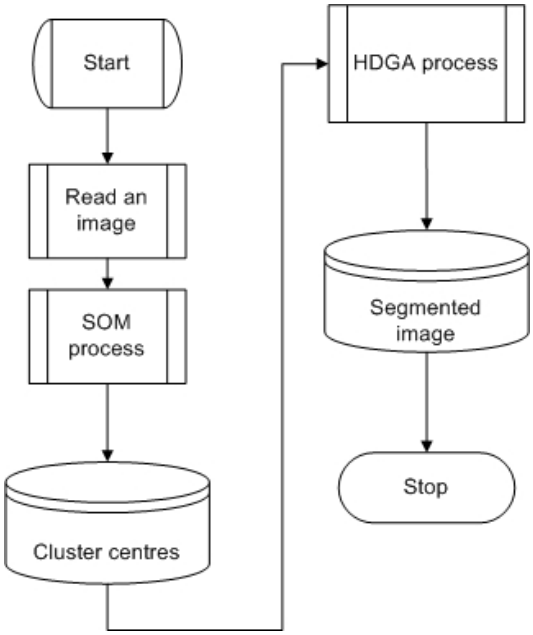


Fig. 4. SOM and HDGA cooperative method

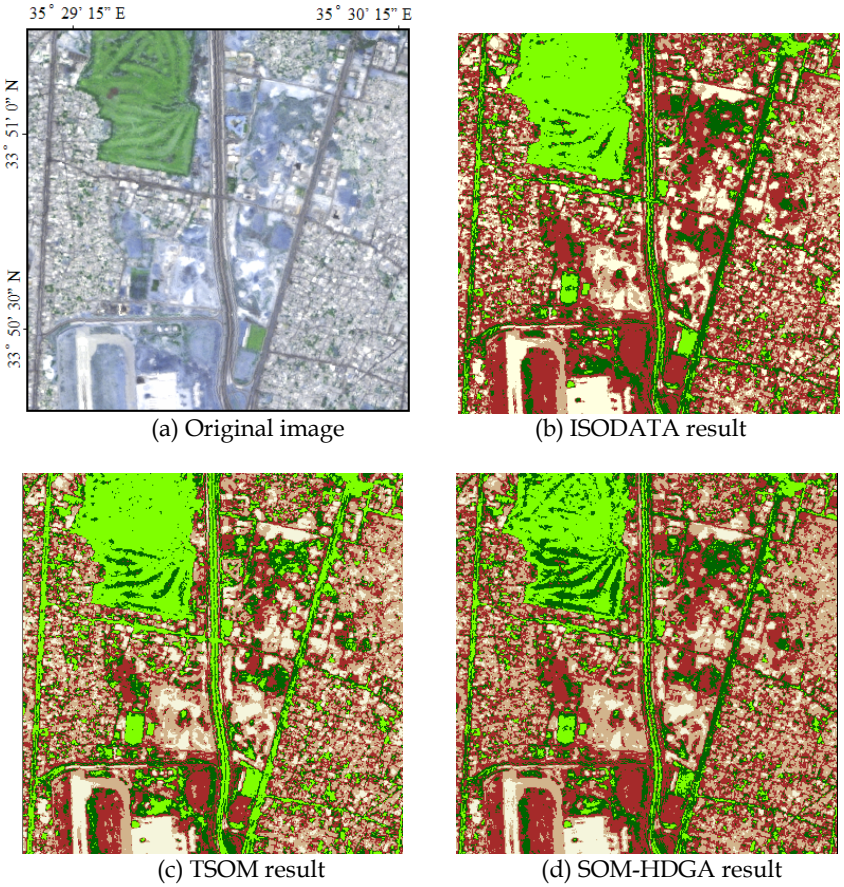
5. Experimental results

The proposed image segmentation methods are implemented using C language on an Intel Centrino™ -1.7 GHz computer. The number of iterations for SOM is 1000, and HDGA will be terminated when the fitness value of the best individual remains unchanged over the

past 20 generations. Three experiments are conducted using two types of satellite images (Spot-5 and Ikonos-II). Spot-5 started orbiting earth in 2002. The image captured by Spot-5 consists of 3 bands, of 5m resolution pan-sharpened by a panchromatic band of 2.5m resolution). Spot-5 is commonly referred to as a pushbroom scanner meaning that all scanning parts are fixed, and scanning is accomplished by the forward motion of the scanner. Ikonos-II is an important high-resolution satellite operated by GeoEye formerly Space Imaging LLC. Its capabilities include capturing a 3.2m multispectral, near-infrared (NIR) and 1 m panchromatic resolution.
The experiments are conducted in order to demonstrate the accuracy, and efficiency of SOM with the clustering technique and cooperating with another method. The results are compared to that of the Iterative Self-organizing Data (ISODATA) algorithm. ISODATA clustering is iterative in that it repeatedly performs an entire classification (outputting a thematic raster layer) and recalculates statistics. Self-Organizing refers to the way in which it locates clusters with minimum user input. The ISODATA method uses minimum spectral distance to assign a cluster for each candidate pixel. The process begins with a specified number of arbitrary cluster means or the means of existing signatures, and then it processes repetitively, so that those means shift to the means of the clusters in the data. The ISODATA algorithm has some further refinements by splitting and merging of clusters (Jensen, 1996).
At the end of the segmentation processes, several samples are collected from the segmented images representing different major classes. These samples are verified using a Global Positioning System (GPS) device with high accuracy (2 meter) and confusion matrices (Kohavi & Provost, 1998) which contain information about actual and predicted segmentation done by a segmentation method. The actual values are represented in the columns and the predicted values are represented in the rows of the matrix. Performance of such systems is commonly evaluated using the data in the matrix. Samples are selected from the segmented image based on the size of the area and the ambiguity of the results. The bigger the area or the more ambigξous the more samples are collected, and the smaller or the clearer the area the less samples are collected.
It is well known that when the ratio of the number of training samples to the number of feature measurements is small, the estimates of the discriminant functions are not accurate this is called Hughes phenomenon (Shahshahani & Landgrebe, 1994).

5.1 Spot V image segmentation

The first test image is a Spot V image with size of 360×360 pixels (see Figure 5(a)). The image represents an area near an airport. ISODATA segmentation result is shown in Figure 5(b), TSOM segmentation result is shown in Figure 5(c), and SOM-HDGA segmentation result is shown in Figure 5(d). In order to compare these experimental results of ISODATA, TSOM, and SOM-HDGA segmentation methods, a confusion matrix is used where terrain verification is applied and compared with the results of these segmentation methods. This process is accomplished using four different classes (1= Vegetation type I (dark green), 2= Vegetation type II (light green), 3= Soil (brown color), 4= Urban (tan)) and 400 survey points (see Tables 1, 2, and 3 respectively). The accuracies of the three different segmentation methods are 78 %, 84 %, and 90% respectively.



(a) Original image (b) ISODATA result

(c) TSOM result (d) SOM-HDGA result

Fig. 5. Spot V satellite image and the results of the different segmentation methods

Ground truth classes	1	2	3	4	Total
1- Vegetation– type 1	80	10	2	8	100
2- Vegetation– type 1I	5	78	5	12	100
3-Soil	7	7	80	6	100
4- Urban	12	9	4	75	100
Total	104	104	91	101	400

Table 1. Confusion matrix for ISODATA

Ground truth classes	1	2	3	4	Total
1- Vegetation– type 1	84	7	2	7	100
2- Vegetation– type 1I	2	83	5	10	100
3-Soil	5	6	83	6	100
4- Urban	6	4	5	85	100
Total	97	100	95	108	400

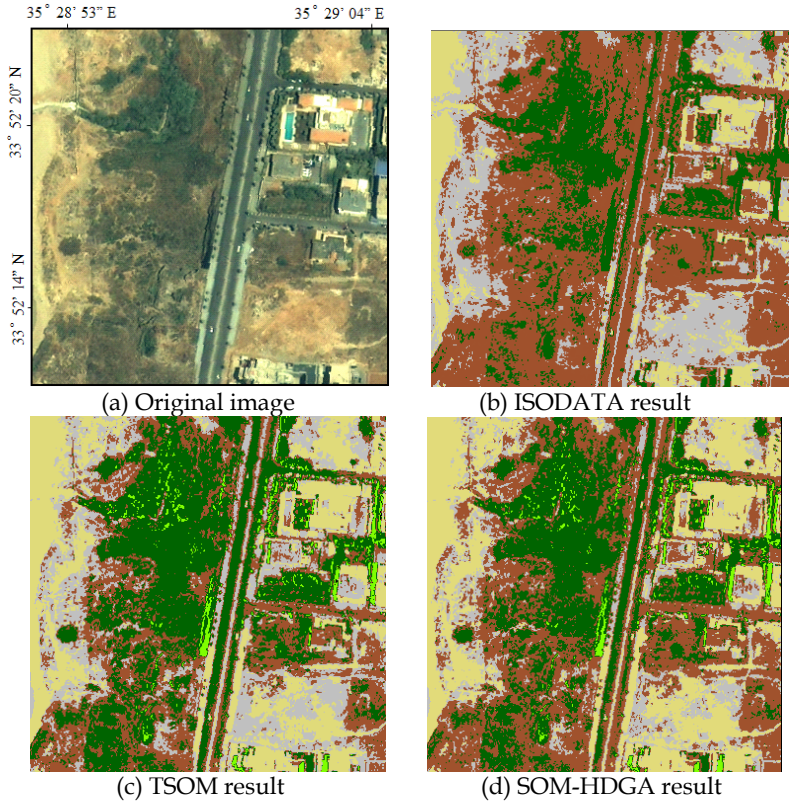
Table 2. Confusion matrix for TSOM

Ground truth classes	1	2	3	4	Total
1- Vegetation– type 1	86	4	4	6	100
2- Vegetation– type 1I	1	91	2	6	100
3-Soil	1	3	94	2	100
4- Urban	3	4	5	88	100
Total	91	102	105	102	400

Table 3. Confusion matrix for SOM-HDGA

5.1 Ikonos-II image segmentation

The second test image is an Ikonos-II image with size 360×360 pixels. This image represents a coastal area in the Mediterranean Basin. The image is selected because of the diversity of classes (urban of different types and structures, vegetation, sand, soil…etc) which creates a complex texture. In addition, the goal is to minimize the effects of the existing of shadows which are the main problem in high resolution images. The image consists of the visible bands pan-sharpened with the panchromatic band (see Figure 6(a)). ISODATA segmentation result is shown in Figure 6(b), TSOM segmentation result is shown in Figure 6(c), and SOM-HDGA segmentation result is shown in Figure 6(d).



(a) Original image (b) ISODATA result

(c) TSOM result (d) SOM-HDGA result

Fig. 6. Ikonos-II satellite image and the results of different segmentation methods

One can notice easily that shadow is mixed with vegetation in ISODATA while it is well separated in the other two segmentation methods (light green). In addition, ISODATA is not separating vegetation properly as compared to the other two methods. Again, the same procedure is used to verify the results of the three segmentation methods. Four classes are selected (1- high to medium dense vegetation “dark green”, 2- Sparse vegetation “brown”, 3- Sand type I and urban "gray", 4- Sand type II and urban "light yellow"). The total number of samples for the 4 classes is 320 distributed evenly between the classes. However, some areas are surveyed extensively in order to clarify the ambiguity which are found in the results (i.e. the segmentation of sandy and shrubs areas are clearly different between the three methods).
The confusion matrices for the results of the three segmentation methods are shown in table 4, 5, and 6 respectively.

Ground truth classes	1	2	3	4	Total
1- Vegetation– type 1	51	20	5	4	80
2- Vegetation– type 1I	22	49	7	2	80
3- Sand type I/urban	2	3	60	15	80
4- Sand type II/urban	10	5	13	52	80
Total	85	77	85	73	320

Table 4. Confusion matrix for ISODATA

Ground truth classes	1	2	3	4	Total
1- Vegetation– type 1	69	7	3	1	80
2- Vegetation– type 1I	8	65	4	3	80
3- Sand type I/urban	1	4	68	7	80
4- Sand type II/urban	1	2	5	72	80
Total	79	78	80	83	320

Table 5. Confusion matrix for TSOM

Ground truth classes	1	2	3	4	Total
1- Vegetation– type 1	72	4	3	1	80
2- Vegetation– type 1I	5	70	3	2	80
3- Sand type I/urban	1	3	71	5	80
4- Sand type II/urban	1	1	3	75	80
Total	79	78	80	83	320

Table 6. Confusion matrix for SOM-HDGA

The accuracies of the three segmentation methods are computed from the matrices and they are 66% for ISODATA, 86 % for TSOM, and 90% for SOM-HDGA.

6. Conclusion and future work

Segmentation is an important step in image processing. The lack of an efficient unsupervised non-parametric method to segment any type of satellite images specifically

high resolution images led us to implement two different sequential methods for satellite image segmentation. These methods are TSOM and SOM-HDGA. When applying these methods to satellite images, the results are more robust and efficient than those obtained with ISODATA segmentaton/classification method. SOM-HDGA overall average efficiency is equal or greater than 90%. However the efficiency of ISODATA degrades with the increase of image resolution. The overall average of ISODATA is about 66 %. SOM-HDGA process robustness and efficiency are due to the fact that a primary solution is provided by SOM to HDGA which in turn uses only the best organized data by avoiding the use of small clusters.

On the other hand, in ISODATA an initial number of clusters must be given for the technique to work. SOM-HDGA performance can be improved using parallel cooperation with more segmentation methods such as Fuzy C-Means. Another approach to improve the segmentation process is to use a knowledge base which contains the cluster centers of previous segmentation on different satellite images with similar date. This is very important for multi-resolution satellite image segmentation. In addition, the knoweledge base may have information about not only the capturing date of the image, but other related information, which will help in speeding up the process of segmenting similar satellite images. This information may be about fixed and variable classes e.g. Forests for fixed classes and arable land for variable classes.

7. References

Aria, E.; Saradjian, M., Amini, J. & Lucas, C. (2004). Generalized Cooccurence matrix to classify IRS-1D images using Neural Network, *Proceedings of XXth ISPRS Congress*, pp. 117-123, Turkey.

Awad, M.; Chehdi, K. & Nasri, A. (2007). Multi-component image segmentation using Genetic Algorithm and Artificial Neural Network. *IEEE Geosciences and Remote Sensing Letters*, Vol. 4, no. 4, pp. 571-575, 2007.

Awad, M.; Chehdi, K. & Nasri, A. (2009a). Multi-component Image Segmentation Using a Hybrid Dynamic Genetic Algorithm and Fuzzy C-Means. *IET image processing*, Vol. 3, No. 2, pp. 52-62.

Awad, M.; Chehdi, K. & Nasri, A. (2009b). Satellite Image Segmentation: a Comparative Analysis between SOM & HGA. I*nternational Journal of Remote Sensing*, Vol. 30, No. 3, pp. 595-610.

Benediktsson, J.; Swain, P., Ersoy, O. & Hong, D. (1990). Neural Network Approaches Versus Statistical Methods in Classification of Multisource Remote Sensing Data. *IEEE Transactions on Geoscience and Remote Sensing,* Vol. 28, No. 4, pp. 540-551.

Jensen, J. (1996). *Introductory Digital Image Processing: A Remote Sensing Perspective*. Englewood Cliffs, New Jersey: Prentice-Hall.

Kohavi, R. & Provost, F. (1998). Glossary of Terms. *Special Issue on Applications of Machine Learning and the Knowledge Discovery Process,* Vol. 30, No. 2/3.

Kohenen, T. (2001). Self-Organizing Maps. *Springer Series Information Sciences*, Vol. 30, 501 pages.

Levine, M. (1985). *Vision in man and machine*, McGraw Hill Publ. Co., New York.

Lopes, A.; Nezry, E., Touzi, R. & Laur, H. (1990). Maximum A Posteriori Speckle Filtering and First Order Textural Models in SAR Images, *Poceedings of International Geoscience and Remote Sensing Symposium (IGARSS)*, vol. 3, pp. 2409-2412, Maryland, USA.

Mason, P. and Pieczynski, W. (1993). SEM algorithm and unsupervised segmentation of satellite images. *IEEE Transactions on Geoscience and Remote Sensing*, Vol. 31, pp. 618-633.

Perkins, S.; Theiler, J., Brumby, S., Harvey, N., Porter, R., Szymansk, J. & Bloch, J. (2000). GENIE: A Hybrid Genetic Algorithm for Feature Classification in Multi-Spectral Images, *Proceedings of SPIE Applications and Science of Neural Networks, Fuzzy Systems, and Evolutionary Computation III 4120*, pp 52-62, USA.

Pratt, W. (1991). *Digital Image Processing 2d ed*, John Wiley & Sons, Inc., New York.

Shahshahani, M. & Landgrebe, A. (1994). The effect of unlabeled samples in reducing the small sample size problem and mitigating the Hughes phenomenon. *IEEE Transactions on Geoscience and Remote Sensing*, Vol. 32**,** pp. 1087- 1095.

Tou, J. & Gonzalez, R. (1974). *Pattern Recognition Principles*, Massachusetts: Addison-Wesley Publishing Company.

Xu, B. & Lin, S. (2002). Automatic color identification in printed fabric images by a fuzzy-neural network. *AATICC Review*, Vol. 2, No. 9, pp. 42-45.

Yao, K.; Mignotte, M., Collet, C., Galerne, P. & Burel, G. (2000). Unsupervised segmentation using a self-organizing map and a noise model estimation in sonar imagery. *Pattern Recognition Letters*, Vol. 33, No. 9, pp. 1575-1584.

Yin, H. & Allinson, N. (1995). On the distribution and convergence of feature space in self-organizing maps. *Neural Computation*, Vol. 7, No. 6, pp. 1178-1187.

Zhang, P.; Verma, B. & Kumar, K. (2003). Neural vs Statistical Classifier in Conjunction with Genetic Algorithm Feature Selection in Digital Mammography, *Proceedings of IEEE* Congress on Evolutionary Computation, CEC'03, pp. 634- 639, Australia.

Zhou, Z.; Wei, S., Zhang, X. and Zhao, X. (2007). Remote sensing image segmentation based on self-organizing map at multiple-scale, Proceedings of SPIE Geoinformatics: Remotely Sensed Data and Information 6752, USA.

16

Bridging the Semantic Gap using Human Vision System Inspired Features

Gaëtan Martens, Peter Lambert and Rik Van de Walle
Multimedia Lab - Ghent University - IBBT
Belgium

1. Introduction

In the last decade, digital imaging has experienced a worldwide revolution of growth in both the number of users and the range of applications. The amount of digital image content produced on a daily basis is still increasing drastically. As from the very beginning of photography, those who took pictures tried to capture as much information as possible about the photograph and in today's digital age, the need for appending metadata is even bigger. However, it is obvious that manually annotating images is a cumbersome, time consuming and expensive task for large image databases, and it is often subjective, context-sensitive and incomplete. Furthermore, it is difficult for the traditional text-based methods to support a variety of task-dependent queries solely relying on textual metadata since visual information is a more capable medium of conveying ideas and is more closely related to human perception of the real world. The dynamic image characteristics require sophisticated methodologies for data visualization, indexing and similarity management and, as a result, have attracted significant research efforts in providing tools for content-based retrieval of visual data. Content-based image retrieval uses the visual contents of an image such as color, shape, texture, and spatial layout to represent and index the image.
Early content-based image retrieval systems were based on the search for the best match to a user-provided query image or sketch (Flickner et al., 1995; Mehrotra et al., 1997; Laaksonen et al., 2002). Such systems decompose each image into a number of low-level visual features (e.g., color histograms, edge information) and the retrieval process is formulated as the search for the best match to the feature vector(s) extracted from a query image. However, it was quickly realized that the design of a fully functional retrieval system would require support for semantic queries (Picard, 1995). The basic idea is to automatically associate semantic keywords with each image by building models of visual appearance of the semantic concepts of interest. However, the critical point in the advancement of content-based image retrieval is the semantic gap. The semantic gap is the major discrepancy in computer vision: the user wants to retrieve images on a semantic level, but the image characterizations can only provide a low-level similarity. As a result, describing high-level semantic concepts with low-level visual features is a challenging task. The first efforts targeted the extraction of specific semantics under the framework of binary classification, such as indoor versus outdoor (Szummer & Picard, 1998), and city versus landscape

classification (Vailaya et al., 1998). More recently, efforts have emerged to solve the problem in greater generality through the design of techniques capable of learning semantic vocabularies from annotated training image collections by applying (both unsupervised and semi-supervised) machine learning techniques, e.g. (Duygulu et al., 2002; Feng et al., 2004).
In computer vision one of the traditional goals is the automatic segmentation and interpretation of general digital images of arbitrary scenes. In the literature, certain methods have been proposed to extract the semantics of scenery images using low-level features. Most of these approaches use image partitioning as intermediate step. Wang et al. use a codebook to segment an image based on the statistics of the regions' color and texture features (Wang et al., 2002). At pixel level, color-texture classification is used to form the codebook. This codebook is used in the next stage to segment an image into regions. The context and content of these regions are defined at image level. Zhu et al. partition the image into equally sized blocks and indexes the regions using a codebook whose entries are obtained from the features extracted from a block (Zhu et al., 2000). In (Li & Wang, 2003) a method is described to use 2-dimensional hidden Markov models to associate the image and a textual description. However, most approaches introduced above can not integrate the semantic descriptions into the regions, and therefore cannot support the high-level querying of images. Depalov et al. use a color-texture segmentation algorithm to segment images depicting natural scenes (Depalov et al., 2006). The features of the obtained regions are used as medium level descriptors to extract semantic labels at region level and later at scene level. However, the use of quantized features may result in weaker segmentations. Turtinen and Pietikäinen applied a Self-Organizing Map trained with local binary patterns to classify outdoor scene images (Turtinen & Pietikäinen, 2003). As a means of supervision, the user selects the map nodes with similar appearance and the corresponding samples are retrained using a smaller map in order to reveal if some classes are mixed up in the same node.
Despite all efforts, humans still outperform the best machine vision systems in many aspects. Humans are very good at getting the conceptual category and layout of a scene within a single fixation. So, building a system that emulates the recognition tasks of the cortex is a challenging and attractive idea. However, in computer vision the use of visual neuroscience has often been limited to a tuning of Gabor filter banks (Jain & Farrokhnia, 1991; Clausi & Jernigan, 2000; Zhang et al., 2000). No real attention has been given to biological features of higher complexity so far. Given the fact that the human vision system is best trained to color and texture perception, these low-level features could play an important role in image understanding. Indeed, Renninger and Malik already have concluded that a texture analysis provides useful information for rapid scene identification (Renninger & Malik, 2004). Therefore, the application of the appropriate features is of utter importance.
This chapter deals with the combination of biologically inspired features and Self-Organizing Maps (Kohonen, 2001) for the classification and recognition of real-world textures and the segmentation of textured images. Analogously to the processing principles of the visual cortex, the unsupervised learning capabilities and visualization techniques of a Self-Organizing Map are utilized with the highly efficient color and texture features. The Self-Organizing Maps are particularly well-suited for the combined task of mapping the high-dimensional and non-linear data distribution to a low dimensional plane while conserving the local neighborhood relations for fast and easy-to-use visualization.

The remaining part of this chapter is organized as follows. Section 2 describes the computational model to calculate the biologically inspired texture features and in Section 3 we briefly introduce the basic model of color perception. Section 4 outlines the calculation of the features and explains the data preprocessing with regard to classification. Unsupervised image partitioning experiments on gray-scale and real-life color textures are presented in Section 5. Section 6 describes the automatic interpretation of the obtained image regions and discusses the possible improvements. Final conclusions and future work appear in Section 7.

2. Texture model

Our system is inspired by the standard model of the human visual system (HVS) (Riesenhuber & Poggio, 2003). The standard model summarizes what most visual neuroscientists generally agree on:

- the first few hundred milliseconds of visual processing in primate cortex follows a mostly feed-forward hierarchy for immediate recognition tasks;
- hierarchical build-up of invariances first to position and scale and to the viewpoint, and more complex transformations requiring the interpolation between several different object views;
- in parallel, an increasing size of receptive fields;
- plasticity and learning probably at all stages and certainly at the level of the cortex;
- learning specific to an individual object is not required for scale and position invariance.

In its simplest form, the view based module of the standard model consists of 4 layers of computational units. The first layer of simple cells S1 in the primary visual cortex (also called the striate cortex or V1) represents linear oriented filters followed by an input normalization. Each unit in the next layer (C1) pools the outputs of simple cells of the same orientation but at slightly different positions by using a maximum operation. Each of these units is still orientation selective but more invariant to the scale, similarly to some complex cells. In the next stage signals from complex cells with different orientations but similar positions are combined (in a weighted sum) into simple cells S2 to create neurons tuned to a dictionary of more complex features. The final layer of C2 units is similar to the C1 cells: by pooling together signals from S2 cells of the same type but at slightly different scales, the C2 units become more invariant to the scale but preserve feature selectivity.

2.1 Simple cells

In a first stage, responses are obtained by applying a Gabor filter bank to an input image. As proposed by (Daugman, 1985) the following family of 2-dimensional isotropic Gabor filters are used to model the receptive cells of the HVS:

$$g_{\lambda,\theta,\varphi}(x,y) = \frac{1}{2\pi\sigma^2}\exp\left(-\frac{x'^2+\gamma^2 y'^2}{2\sigma^2}\right)\cos\left(2\pi\frac{x'}{\lambda}+\varphi\right) \quad (1)$$

where $x' = x\cos\theta - y\sin\theta$ and $y' = x\sin\theta + y\cos\theta$.

The orientation of the filter is represented by θ and σ denotes the standard deviation of the Gaussian which determines the size of the receptive field of the HVS. The phase offset φ

affects the symmetry of the function. The parameter λ determines the spatial wavelength of the receptive field function (1). Since σ and λ are not independent, the standard deviation is selected to satisfy $\frac{\sigma}{\lambda} = 0.56$ in order to obtain a one-octave spatial frequency bandwidth (Kruizinga & Petkov, 1999). Finally, the parameter γ, also called the spatial aspect ratio, affects the receptive field ellipse. It has been found that γ ranges between 0.23 and 0.92 (Jones & Palmer, 1987) and is set to 0.5 according to (Kruizinga & Petkov, 1998).
The response of the receptive field function to an input image its luminance channel $I(x, y)$ is defined by:

$$r_{\lambda,\theta,\varphi}(x, y) = \iint I(s, t) g_{\lambda,\theta,\varphi}(x - s, y - t) ds dt \tag{2}$$

The response $s_{\lambda,\theta,\varphi}(x, y)$ of a simple cell of the visual cortex, modeled by a receptive field function $g_{\lambda,\theta,\varphi}(x, y)$ to $I(x, y)$, is given by:

$$s_{\lambda,\theta,\varphi}(x, y) = \begin{cases} 0 & \text{if } a_\lambda(x, y) = 0 \\ \chi\left(\frac{\frac{r_{\lambda,\theta,\varphi}(x,y)}{a_\lambda(x,y)} R}{\frac{r_{\lambda,\theta,\varphi}(x,y)}{a_\lambda(x,y)} + C} \right) & \text{otherwise} \end{cases} \tag{3}$$

where R denotes the maximum response level, the average gray value $a_\lambda(x, y) = \iint I(s, t) \exp \frac{(x-s)^2 + \gamma^2 (y-t)^2}{2\sigma^2} ds dt$, C is the semi-saturation constant, and $\chi(t) = t$ for $t \geq 0$ and $\chi(t) = 0$ for $t < 0$ (Kruizinga & Petkov, 1998).

2.2 Grating cells

The next layer corresponds to complex cells which provide some tolerance to shift and size. This tolerance is obtained by taking a maximum across neighboring scales and nearby pixels. Grating cells are orientation selective cells which respond strongly to gratings of appropriate periodicity and orientation, but in contrast to the simple cells or some other complex cells, they do not respond to a single bar (Kruizinga & Petkov, 1999). The computational model of a grating cell consists of two stages:

(i) the calculation of the activity of a grating subunit $q_{\lambda,\theta}(x, y)$ with a preferred orientation θ and frequency $1/\lambda$, see (4)

(ii) the summation of the responses for a given θ, see (6).

A grating subunit $q_{\lambda,\theta}(x, y)$ takes as input the simple cell outputs defined in (3) and will be activated if there are at least 3 parallel bars with orientation θ and frequency $1/\lambda$:

$$q_{\lambda,\theta}(x, y) = \begin{cases} 1 & \text{if } \forall n, M_{\lambda,\theta,n}(x, y) \geq \rho N_{\lambda,\theta}(x, y) \\ 0 & \text{if } \exists n, M_{\lambda,\theta,n}(x, y) < \rho N_{\lambda,\theta}(x, y) \end{cases} \tag{4}$$

where $0 < \rho < 1$ is a threshold in the proximity of 1. As suggested by (Kruizinga & Petkov, 1998), it is assigned 0.9. The quantities $M_{\lambda,\theta,n}$ and $N_{\lambda,\theta}$ are computed as follows:

$$\begin{cases} M_{\lambda,\theta,n}(x, y) = \max\{s_{\lambda,\theta,\varphi_n}(u, v)\} \\ N_{\lambda,\theta}(x, y) = \max\{M_{\lambda,\theta,n}(x, y) | n = -3, -2, -1, 0, 1, 2\} \end{cases} \tag{5}$$

where $\varphi_n = 0$ for $n = \{-3, -1, 1\}$ and $\varphi_n = \pi$ for $n = \{-2, 0, 2\}$. Finally, u and v satisfy the condition:

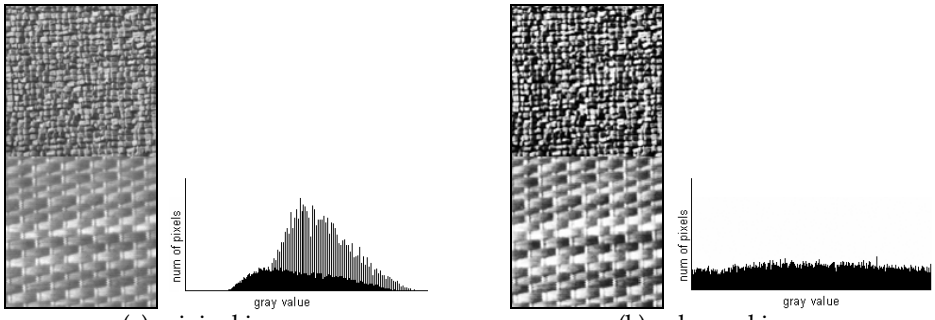
$$\begin{cases} n\frac{\lambda}{2}\cos\theta \leq u - x < (n+1)\frac{\lambda}{2}\cos\theta \\ n\frac{\lambda}{2}\sin\theta \leq v - y < (n+1)\frac{\lambda}{2}\sin\theta \end{cases} \tag{6}$$

A grating subunit will be activated $(q_{\lambda,\theta}(x,y) = 1)$ if for the preferred orientation θ and spatial frequency $1/\lambda$, the receptive field function (2) is alternately activated in intervals of length $\lambda/2$ for $n = -3, -2, .., 2$ and this along a line segment of length 3λ centered on point (x, y). In other words, the condition is fulfilled in case there are at least 3 parallel bars with spacing λ and orientation θ of the normal encountered. In the final stage, the output of the grating cell operator $w_{\lambda,\theta}$ is computed as:

$$w_{\lambda,\theta}(x,y) = \frac{1}{\sqrt{2\pi\sigma}} \iint (q_{\lambda,\theta}(s,t) + q_{\lambda,\theta+\pi}(s,t)) \exp\left(-\frac{(x-s)^2 + (y-t)^2}{2(5\sigma)^2}\right) dsdt \tag{7}$$

2.3 Enhanced grating cell operator

This operator first applies a histogram equalization on the original input image $I(x, y)$ to obtain the enhanced image $\overline{I}(x, y)$. Histogram equalization employs a monotonic, non-linear mapping which re-assigns the intensity values of pixels in the input image such that the intensity values of the output image are more uniformly distributed (*i.e.* a flat histogram) and the image has a higher contrast (see Fig. 1).



(a) original image (b) enhanced image

Fig. 1. Luminance histograms of the original and enhanced image

For each gray level j in the input image $I(x, y)$, the new value $\overline{j}$ is calculated as follows:

$$\overline{j} = l \sum_{i=0}^{j} \frac{n_i}{n} \tag{8}$$

where l is the maximum gray level, n the total number of pixels and n_j the number of occurrences of gray level j in $I(x, y)$. The result of this operation is that $\overline{I}(x, y)$ has not only a higher contrast, but also its details are enhanced such that the salient texture specific periodicities are more distinguishable. The enhanced grating cell features $\overline{w}_{\lambda,\theta}(x, y)$ are obtained by substituting the enhanced image $\overline{I}(x, y)$ in equations (2) and (3). As shown in our previous work (Martens et al., 2007), the application of the enhanced grating cell operator has a positive influence on texture classification results.

3. Color Perception

Since color is the primary visual stimulus, the choice of a color system is of great importance for the purpose of proper image retrieval. Color can be modeled and interpreted in many different ways and color systems have been developed for various purposes, such as *RGB* & *CMYK* for displaying and printing, *YIQ* & *YUV* for television and video transmission efficiency, *XYZ* for color standardization, etc.

The first geometrical model of color perception was created in the 17th century by Isaac Newton. He epitomized his experiments in light and pigment mixing by ingeniously overlapping the red and violet ends of the spectrum to create a hue circle. This circle shows the spectrum as a continuous gradation of color from red to violet, and from violet to red via the mixed colors carmine, magenta and purple. This circular representation of color is also used in the HSI space, where HSI stands for *hue, saturation,* and *intensity* (Gevers, 2001).

In the HVS color vision is mediated by specialized nerve cells in the retina, called cones. The ability to discern different wavelengths of light (*i.e.* colors) gives us more information for detecting and identifying objects than would be provided solely by black and white vision. The human retina has three types of cones which makes color detection possible: red, green and blue cones. By appropriately mixing these three primary colors it is possible to match all of the colors in the visible spectrum. The latter observation is known as the trichromatic theory (von Helmholtz, 1867).

However, the fact that some colors cannot be perceived in combination, e.g. "reddish green" or "bluish yellow", cannot be explained by the trichromatic theory. This proved to Edward Hering that the visual substances were organized as opponent processes (Herring, 1874). In summary, Hering proposed there are six fundamental color processes arranged as three visual contrasts including two opponent processes:

(i) black versus white,
(ii) red - green opponent process,
(iii) blue - yellow opponent process.

By the middle of the 20th century it was proven that both theories are necessary to explain the physiological processes of color perception. So, color vision is a dual process: the trichromatic theory is correct at photo-pigment level (by conical photoreceptors in the retina) and the opponent theory is correct at the neural level (by opponent cells found in the lateral geniculate nucleus).

4. Features

Scaling of the feature vectors is of special importance since the Self-Organizing Map classifier uses the Euclidean metric to measure the distances between feature vectors, otherwise bigger variables tend to dominate the others. The sigmoidal transformation (also called the softmax transformation) has been applied since it reduces the influence of outliers in the data. We also empirically observed that this normalization gives the best results, i.e.: $x_i' = 1/(1+e^{\widehat{x_i}})$ where $\widehat{x_i} = (x_i - \overline{x})/\sigma_x$ for a vector x with mean $\overline{x}$ and standard deviation σ_x.

4.1 Texture

The texture features are obtained by combining enhanced grating cell features (see Section 2.3) with spatially smoothed Gabor responses. The latter are obtained by convolving the simple cell responses, see (2) where $\varphi = 0$, with a Gaussian with standard deviation 2σ. Smoothed Gabor responses are known to improve the performance for texture analysis (Bovik et al., 1990). The frequencies for the filters are $\sqrt{2}, 2\sqrt{2}, 4\sqrt{2}, 8\sqrt{2}$, and $16\sqrt{2}$ cycles per image and we use eight orientations (θ=0, $\frac{\pi}{8}$,..,$\frac{7\pi}{8}$). This results in an 80-dimensional vector (smoothed Gabor responses + enhanced grating cell responses) to represent a texture feature.

4.2 Color

Digital images are mainly stored in RGB and can thus easily be transformed into color opponent values (COV) using the following transformation:

$$\begin{bmatrix} RG \\ BY \\ KW \end{bmatrix} = \begin{bmatrix} 1/2 & -1/2 & 0 \\ -1/4 & -1/4 & 1/2 \\ 1/3 & 1/3 & 1/3 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} \tag{9}$$

where RG, BY and KW represent the red-green, blue-yellow and black-white channel, respectively. For R, G, and B values between 0 and 255, the values for RG and BY range between -127.5 and 127.5, while KW ranges between 0 and 255. Remark that the transformation from RGB to HSI is computationally more expensive than the transformation into COV. Both the HSI and COV color space are considered in our experiments.

5. Unsupervised segmentation

Texture segmentation experiments are applied on multiple composite images of 256×256 and 512×512 pixels containing gray-scale textures from the Brodatz album (Brodatz, 1966). No pixel adjacency information has been used in this clustering process. Introducing pixel adjacency information generally boosts the classification correctness due to the fact that pixels belonging to the same texture are close to each other, and consequently, they should be clustered together. However, the latter requires a priori information about the image (which is not always available, e.g., in digital photos). Consequently, this method will not perform well if some texture regions are not adjacent in the image.

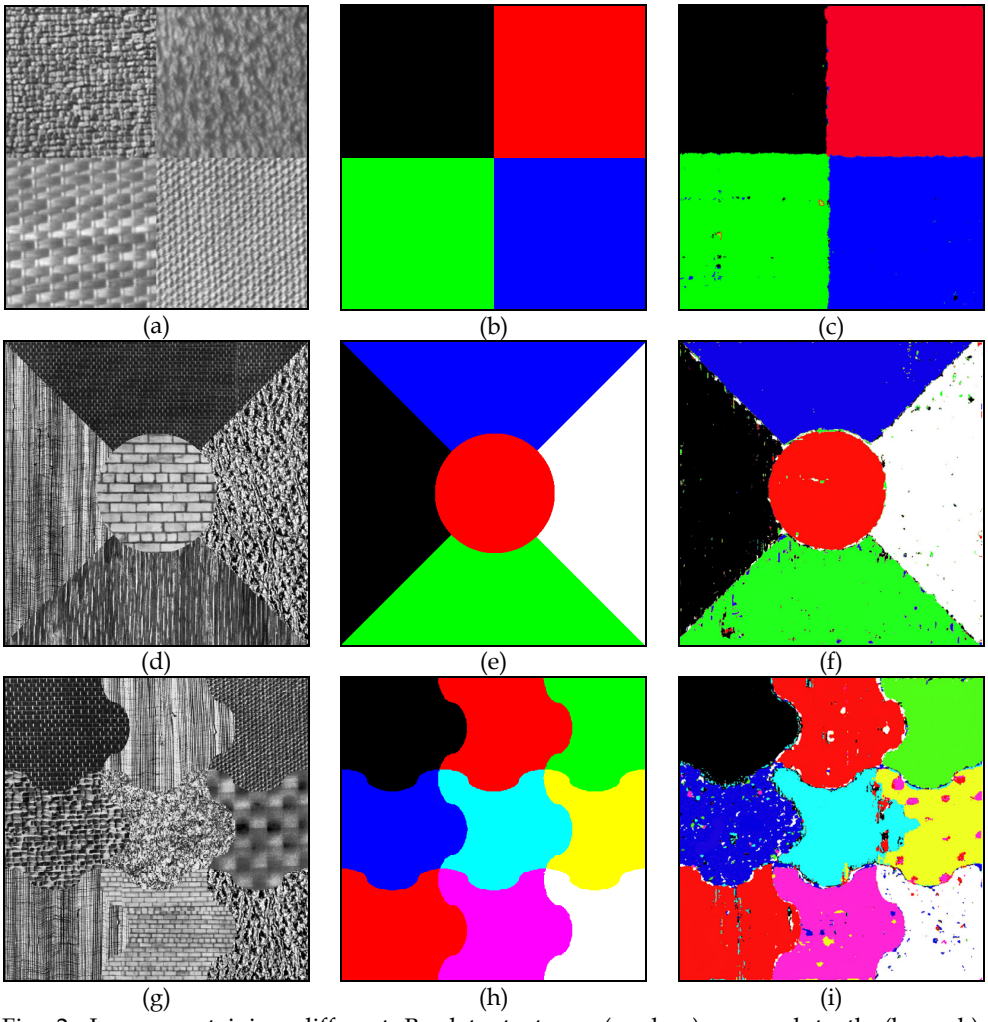
To segment an image, the extracted texture features (see Section 4.1) are employed to train a 2-dimensional Self-Organizing Map (SOM). In a first stage the map is linearly initialized

along the 2 greatest eigenvectors of the data. Next, the SOM is trained using the well-known batch-training algorithm which is, in contrast to the sequential training algorithm, much faster to calculate and the results are as just as good. A result of the training process is that pixels belonging to the same texture are assigned to the same or adjacent nodes. The number of nodes has evidently an influence on the classification result. A general rule is that a higher number of nodes results in better classification results but a side effect is that overclassification may occur. On the other hand, small-sized maps are more attractive because of their lower computational cost during the training phase. Nevertheless, we are able to use small sized maps to receive decent segmentation results because of the distinguishing characteristics of the proposed texture features. For the classification of images containing 2, 4, 5, and 9 textures, maps of dimension 4×2, 4×4, 8×7, and 8×9 nodes are trained, respectively. Figure 2 exemplifies the segmentation of images containing 4, 5 and 9 Brodatz textures using enhanced grating cell and smoothed Gabor features. Table 1 depicts the precision of the segmentation using real Gabor filter responses, smoothed real Gabor filter responses, enhanced grating cell responses, and the combination of the latter and former features. We notice that the real Gabor filter responses have low discriminating capabilities compared to the other features. The SOM-based classifier produces clearly the most precise segmentation using the combination of the enhanced grating cell features with smoothed Gabor filter responses.

# *textures*	*Real Gabor*	*Smoothed Gabor*	*Enhanced grating cell*	*Enhanced grating cell + smoothed Gabor*
2	0.69	0.97	0.89	0.97
4	0.34	0.85	0.72	0.90
5	0.42	0.85	0.67	0.89
9	0.25	0.81	0.59	0.86

Table 1. Segmentation precision of Brodatz textures.

To investigate the application of the proposed color and texture features for segmenting scenery images, experiments are conducted on 100 composite images of size 512 × 512 pixels containing randomly selected natural color textures. Each collected texture belongs to one of these five classes: (i) *bricks*, (ii) *grass*, (iii) *tree*, (iv) *sky*, and (v) *water*, as exemplified in Fig. 3. An example of the partitioning of natural textures is shown in Fig. 4. It is important to remark that in contrast to the gray-scale textures from the Brodatz album, the intra-variation in terms of orientation and scale of a natural texture class in scenery images is much higher. The latter is exemplified in Fig. 5 which consists of four different grass textures. As can be seen, it is even for the human eye hard to distinct the upper two textures, but the difference with the grass textures at the bottom of the image is much larger. Nevertheless, our segmentation algorithm is, to a certain extent, still capable to distinguish them, even using such a small-sized SOM. The segmentation results depicted in Table 2 are obtained by applying a 4×4 SOM for unsupervised segmentation. As can be seen, the combination of color and texture features gives a relatively stable segmentation result. We also notice that the application of color features gives a slight boost of about 5% while the precision using solely color rapidly decreases. The HSI and COV color space induces almost identical segmentation results. Since the transformation of RGB into COV is computational less expensive than HSI, the COV color space will be used in the next stage of our experiments.



(a) (b) (c)

(d) (e) (f)

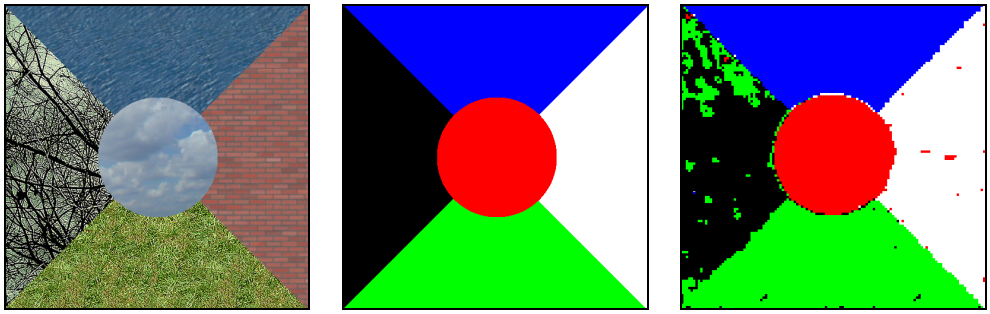
(g) (h) (i)

Fig. 2. Image containing different Brodatz textures (a, d, g); ground truth (b, e, h); segmentation using enhanced grating cell and smoothed Gabor features (c, f, i)

Fig. 3 Examples of real life textures: *branches* (a, b); *bricks* (c, d), *grass* (e), *sky* (f, g) and *water* (h, i).

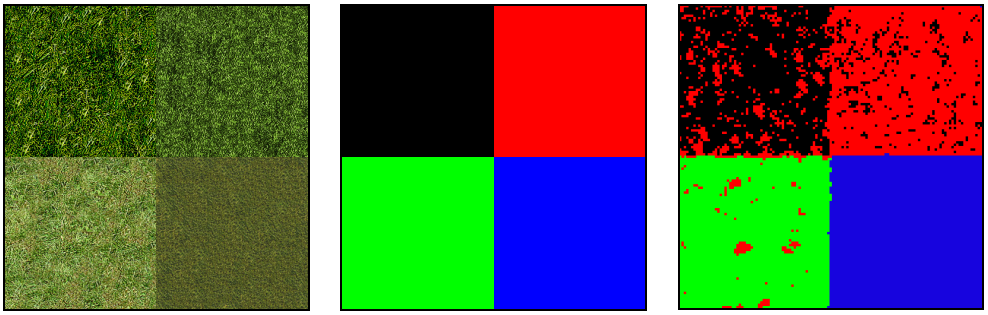
# *textures*	*COV*	*HSI*	*Texture*	*COV + Texture*	*HSI + Texture*
4	0.75	0.76	0.89	0.92	0.89
5	0.74	0.69	0.89	0.94	0.93
9	0.53	0.49	0.83	0.91	0.91

Table 2. Segmentation precision of real-life color textures



(a) natural textures (b) ground truth (c) segmentation

Fig. 4. Example of the segmentation of five natural textures.



(a) grass textures (b) ground truth (c) segmentation

Fig. 5. Segmentation of similar grass textures using COV + texture features.

6. Labeling of image regions

Using the previously computed color-texture samples, a 10×10 SOM is trained such that similar vectors are grouped in the same or in adjacent nodes. After this training phase, each SOM node is assigned a 5-dimensional probability vector v_c by counting the labels of the corresponding training vectors such that $v_c(i)$ is the probability that the label of node c is i. To test the labeling behavior, we employ ten random scenery images from the World Wide Web (containing no other classes than those listed above). To label a segment of an image, the following procedure is applied. At first, the best matching unit c_j for each corresponding feature vector is calculated. Then, the probabilities $v_{c_j}(i)$ are summed together for each i. Figure 6 shows a 2-dimensional mesh visualization of the 10×10 SOM along the 2 highest principle components of the training vectors. The color of each node in Fig. 5 corresponds to the label l with the highest probability, i.e. $v_c(l) > v_c(i), i \neq l$, and illustrates the fact that the different texture classes are nicely clustered.

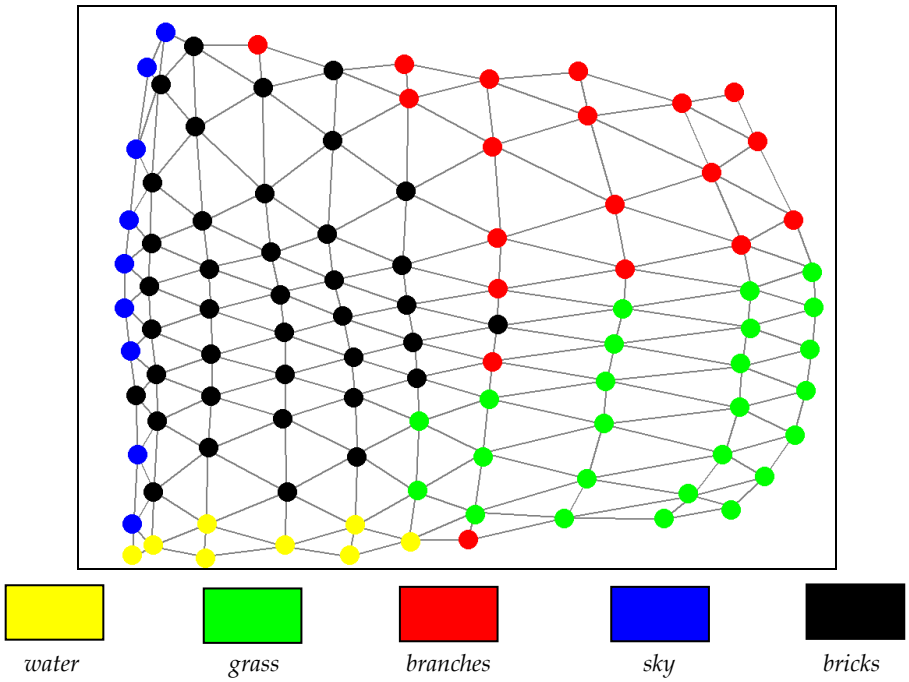


Fig. 6 A 10×10 SOM trained with natural color texture features. The nodes are given the label with the highest probability

Finally, the label of an image partition is obtained by selecting the label k with the highest probability, i.e. $k = max_i \sum_j v_{c_j}(i)$. Using the in Section 4 proposed texture and COV features, the average precision of the classification is 89% of the pixels while the recall is 86%. The labeling precision without an intermediate segmentation step was 72% (Martens et al., 2008). Remark that the ground-truth images are created manually and therefore they should be interpreted as an approximation rather than a certainty. The interpretation of some scenery images is exemplified in Fig. 7. In this figure, different types of errors occur. At first, isolated pixels and edges are misclassified. The latter occurs because they are assigned to the same cluster of nodes in the segmentation step. Consequently, during the labeling step, they will be given the wrong label. To avoid this, small blobs can be filtered out by assigning thresholds to define the minimum size of a partition. Information of enclosing or adjacent regions can then be used to find the most probable label. Another type of error is caused by misclassification in the labeling step. An example hereof is the wrong interpretation of the roof, see Fig. 7 (i). This is mainly caused by the fact that the training set of the class *bricks* doesn't contain any examples which are similar to the corresponding region in Fig. 7 (g). The *sky* blob in the lake of Fig. 7 (d) is also a fault. However, the latter is due to reflection: the increase of brightness causes the contrast to decrease such that the texture significantly alters and a misclassification results. Such errors can only be corrected by incorporating domain knowledge such that, e.g., no blob of bricks can 'float' in the sky.

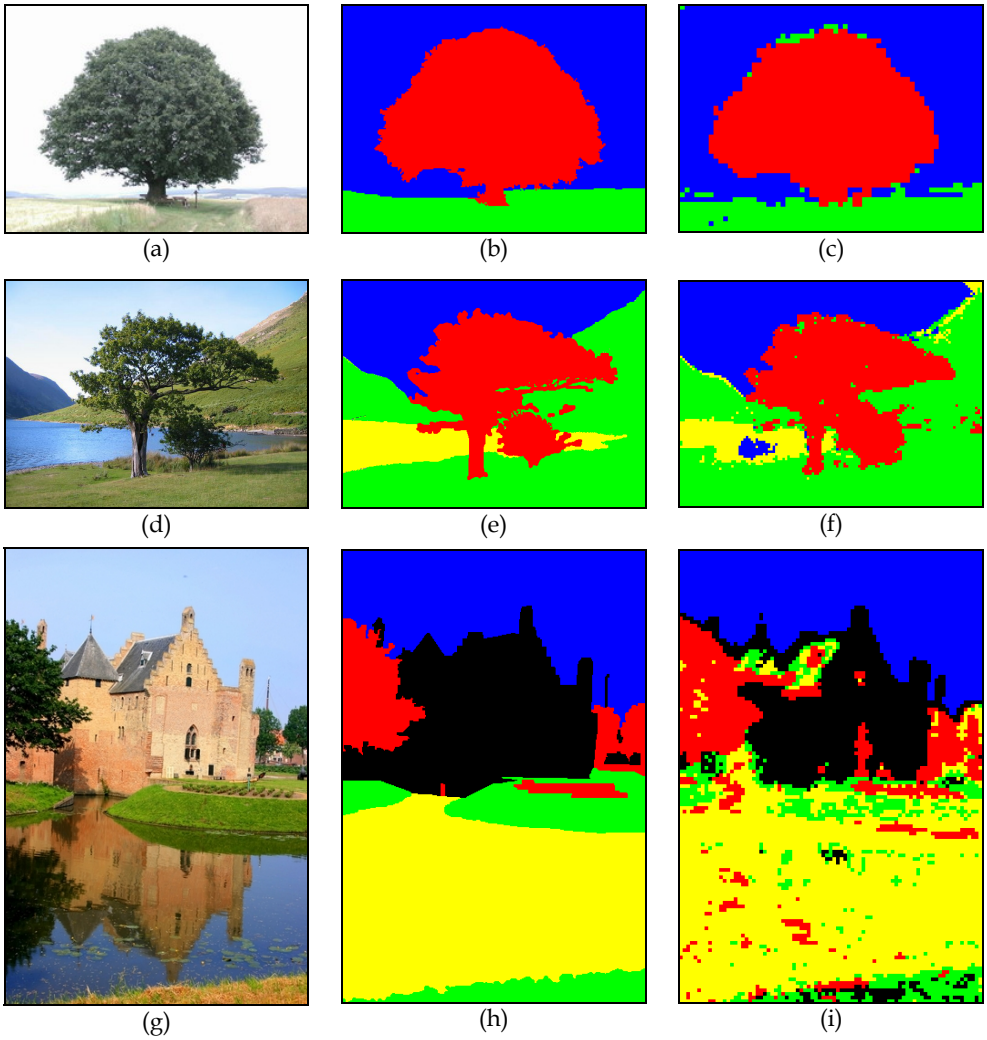


Fig. 7. Interpretation of scenery images: original image (a, d, g), ground truth (b, e, h), and classification (c, f, i).

Other errors, e.g., the *water* detected as *grass* (or vice versa), emerge from the fact that the scaling of certain textures alters due to the perspective. This problem is harder to tackle. Enhancing the segmentation process, e.g. by using a larger SOM, will certainly be helpful but a special approach might be needed for those regions. However, a top-down approach for the detection and correction of misclassifications by embedding domain knowledge is out of scope of this chapter.

7. Conclusion and future work

Due to the semantic gap, the automatic interpretation of images is an intricate task. In this chapter, we have presented a bottom-up approach for the segmentation and interpretation of outdoor scenery images. We established a link between the proposed low-level, biological features and some predefined semantic concepts by applying a SOM for classification. Our method generally consists of two stages. At first, color opponent values and textures are extracted from the image's pixels. Color opponent values induce comparable classification results as colors from the HSI space. Since the transformation of RGB into color opponent values is computationally less expensive than the transformation into HSI, the former are preferred over the latter. The texture features consist of enhanced grating cell features and smoothed Gabor responses and correspond to outputs of cells found in the primary visual cortex of primates and humans. Analogously to the processing principles of the auditory and visual cortex, Self-Organizing Maps are used for the unsupervised segmentation and labeling of textured images. Even using small-sized maps, high precision image segmentations can be obtained (both on gray-scale and on natural color textures). By adding color information, the precision of the segmentation results averagely increased with 5% to a total of 91% of the pixels. In the next stage, the same features are used to train a Self-Organizing Map with textures belonging to one of the 5 predefined classes: (i) *grass*, (ii) *bricks*, (iii) *branches*, (iv) *water*, and (v) *sky*. This map is then used to label the previously obtained image segments. Experiments conducted on randomly collected images from the World Wide Web achieved a precision of 89%. The latter observations indicate that the application of biologically inspired features is very useful for scene interpretation and categorization. We further believe that the classification can be improved by (i) a more accurate segmentation, (ii) a larger, more representative training set, and by (iii) introducing high-level domain knowledge (i.e. a top-down approach). These aspects will be thoroughly investigated. In order to recognize more concepts, we have experienced that when extra texture classes are added to the training set, the number of misclassifications drastically increases. A solution to this problem might be the introduction of hierarchical or tree-structured Self-Organizing Maps (Koikkalainen & Oja, 1990). Nodes containing two (or more) classes are, in a next stage, split up into different clusters what results in the separation of the related concepts. Furthermore, since the visual cortex contains different types of (complex) cells which are tuned to a specific task (e.g., for the detection of edges), we believe that they can also play an important role in image understanding.

8. Acknowledgement

The research activities as described in this paper were funded by Ghent University, the Interdisciplinary Institute for Broadband Technology (IBBT), the Institute for the Promotion of Innovation by Science and Technology in Flanders (IWT), the Fund for Scientific Research-Flanders (FWO-Flanders), and the European Union.

9. References

Bovik A. C., Clark M. & Geisler W. S., (1990), Multichannel Texture Analysis Using Localized Spatial Filters, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12 (1), pp. 55-73.

Brodatz P. (1966), Textures: A Photographic Album for Artists and Designers, Dover, New York.

Clausi D. A., Jernigan M. E. (2000), Designing Gabor Filters for Optimal Texture Separability, *Pattern Recognition*, 33, pp. 1835-1849.

Daugman J. (1985), Uncertainty Relation for Resolution in Space, Spatial Frequency and Orientation Optimized by Two-Dimensional Visual Cortical Filters, *Journal of the Optical Society of America A: Optics, Image Science, and Vision*, 2, pp. 1160-1169.

Duygulu P., Barnard K., Freitas N. & Forsyth D. (2002), Object recognition as machine translation: Learning a lexicon for a fixed image vocabulary, *Proceedings of the European Conference on Computer Vision*.

Feng S., Manmatha R. & Lavrenko V. (2004). Multiple Bernoulli relevance models for image and video annotation, *Proceedings of the IEEE Computer Vision and Pattern Recognition*.

Flickner M., Sawhney H., Niblack W., Ashley J., Huang Q. , Dom B., Gorkani M., Hafner J., Lee D., Petkovic D., Steele D., & Yanker P.(1995), Query by image and video content: The qbic system. *Computer*, 28 (9), pp. 23–32.

Gever T. (2001), Color-Based Retrieval, In: *Principles of Visual Information Retrieval*, Springer, Lew M. S., (Ed.), pp. 11-49, Springer, 1852333812, London, Great Britain.

Hering E. (1874), Zur Lehre vom Lichtsinne.

Jain A. K., Farrokhnia F. (1991), Unsupervised texture segmentation using Gabor filters, *Pattern Recognition*, 24 (12), pp. 1167-1186.

Jones J., Palmer A., (1987), An Evaluation of the Two Dimensional Gabor Filter Model of Simple Receptive Fields in Cat Striate Cortex, *Journal of Neurophysiology*, 58, pp. 1233-1258.

Kohonen T. (2001), *Self-organizing Maps*, Springer, 3540679219, Berlin, Germany.

Koikkalainen P. & Oja E. (1990), Self-organizing hierarchical feature maps, *Proceedings of International Joint Conference on Neural Networks*, 2, San Diego, CA, USA, pp. 279-285.

Kruizinga P. & Petkov N. (1998) Grating Cell Operator Features for Oriented Texture Segmentation, Proceedings of the 14th International Conference on Pattern Recognition, Brisbane, Australia, pp. 1010-1014.

Kruizinga P. & Petkov N., (1999) Nonlinear Operator for Oriented Texture, *IEEE Transactions on Image Processing*, 8 (10), pp. 1395-1407.

Laaksonen J., Koskela M., & Oja E., (2002), PicSOM - Self-Organizing Image Retrieval With MPEG-7 Content Descriptors, *IEEE Transactions on Neural Networks*, 13 (4), pp. 841-853.

Li J. & Wang J. (2003), Automatic linguistic indexing of pictures by a statistical modeling approach, *IEEE Transactions on. Pattern Analyses and Machine Intelligence*, 25 (9).

Martens G., Poppe C., Lambert P. & Van de Walle R. (2008), Unsupervised texture segmentation using biologically inspired features, *Proceedings of the 2008 IEEE 10th Workshop on Multimedia Signal Processing*, pp. 159-164.

Mehrotra S., Rui Y., Ortega-Binderberger M., & Huang T. S. (1997), Supporting content-based queries over images in mars. *Proceedings of IEEE International Conference on Multimedia Computing and Systems*, pp. 632–633.

Picard R., (1995). Digital Libraries: Meeting Place for High-Level and Low-Level Vision, *Proceedings of the Asian Conference on Computer Vision*.

Renninger, L. W. & Malik, J. (2004), When is scene recognition just texture recognition?, *Vision Research*, vol. 44, pp. 2301-2311.

Riesenhuber M. & Poggio T. (2003). How the visual cortex recognizes objects: The tale of the standard model, *The Visual Neurosciences*, 2, pp. 1640-1653.

Szummer M. and Picard R., (1998). Indoor-outdoor image classification, *Proceedings of the Workshop in Content-based Access to Image and Video Databases*, Bombay, India.

Turtinen M. & Pietikäinen M. (2003). Visual training and classification of textured scene images}, *Proceedings of the 3rd International Workshop on Texture Analysis and Synthesis*, pp. 101-106.

Vailaya A., Jain A., and Zhang H. (1998). On image classification: City versus landscape, *Pattern Recognition*, 31, pp. 1921–1936.

von Helmholtz H. (1867), Handbuch der Physiologischen Optik, Leopold Voss, Leipzig.

Wang W., Song Y., and A. Zhang. (2002), Semantics retrieval by content and context of image regions, *Proceedings of the 15th International Conference on Vision Interface*, pp. 17–24, 2002.

Zhang, J., Tan, T., Ma, L. (2002), Invariant texture segmentation via circular Gabor filter, *Proceedings of the 16th IAPR International Conference on Pattern Recognition*, 2, pp. 901–904.

Zhu L., Zhang A., Rao A., & Srihari R. (2000). Keyblock: An approach for content-based image retrieval}, *ACM Multimedia 2000*, pp. 157–166, Los Angeles, CA, 2000.

17

Face Recognition Using Self-Organizing Maps

Qiu Chen, Koji Kotani, Feifei Lee and Tadahiro Ohmi
Tohoku University
Japan

1. Introduction

As an active research area, face recognition has been studied for more than 20 years. Especially, after the September 11 terrorist attacks on the United States, security systems utilizing personal biometric features, such as, face, voice, fingerprint, iris pattern, etc. are attracting a lot of attention. Among them, face recognition systems have become the subject of increased interest (Bowyer, 2004). Face recognition seems to be the most natural and effective method to identify a person since it is the same as the way human does and there is no need to use special equipments. In face recognition, personal facial feature extraction is the key to creating more robust systems.

A lot of algorithms have been proposed for solving face recognition problem. Based on the use of the Karhunen-Loeve transform, PCA (Turk & Pentland, 1991) is used to represent a face in terms of an optimal coordinate system which contains the most significant eigenfaces and the mean square error is minimal. However, it is highly complicated and computational-power hungry, making it difficult to implement them into real-time face recognition applications. Feature-based approach (Brunelli & Poggio, 1993; Wiskott et al., 1997) uses the relationship between facial features, such as the locations of eye, mouth and nose. It can implement very fast, but recognition rate usually depends on the location accuracy of facial features, so it can not give a satisfied recognition result. There are many other algorithms have been used for face recognition. Such as Local Feature Analysis (LFA) (Penev & Atick, 1996), neural network (Chellappa et al., 1995), local autocorrelations and multi-scale integration technique (Li & Jain, 2005), and other techniques (Goudail et al., 1996; Moghaddam & Pentland, 1997; Lam & Yan, 1998; Zhao, 2000; Bartlett et al., 2002 ; Kotani et al., 2002; Karungaru et al., 2005; Aly et al., 2008) have been proposed.

As a neural unsupervised learning algorithm, Kohonen's Self-Organizing Maps (SOM) has been widely utilized in pattern recognition area. In this chapter, we will give an overview in SOM-based face recognition applications.

Using the SOM as a feature extraction method in face recognition applications is a promising approach, because the learning is unsupervised, no pre-classified image data are needed at all. When high compressed representations of face images or their parts are formed by the SOM, the final classification procedure can be fairly simple, needing only a moderate number of labeled training samples. In this chapter, we will introduce various face recognition algorithms based on this consideration.

The chapter will be organized as follows: Section 1 contains an introduction to the chapter. Section 2 presents a review of conventional face recognition currently. Section 3 gives a brief introduction on Self-Organizing Maps (SOM). Section 4 presents in details on various face recognition applications using SOM, including analyses and discussions of their advantages and demerits, and refer to the direction of research of SOM-based face recognition in future. Section 5 gives a conclusion of the chapter.

2. Review of Face Recognition

A number of face recognition algorithms have been proposed (Chellappa et al., 1995; Zhao, 2003; Li et al., 2005; Tan et al., 2006; Abate et al., 2007). These algorithms can be roughly divided into two approaches, namely, structure-based and statistics-based.
In the structure-based approaches (Brunelli & Poggio, 1993; Wiskott et al., 1997; Ben & Nandy, 1998; Lam & Yan, 1998; Li & Lu, 1999), recognition is based on the relationship between human facial features such as eye, mouth, nose, profile silhouettes and face boundary. Statistics-based approaches (Turk & Pentland, 1991; Zhao, 2000) attempt to capture and define the face as a whole. The face is treated as a two dimensional pattern of intensity variation. Under this approach, the face is matched through finding its underlying statistical regularities.

2.1 Structure-based Face Recognition Algorithms

In Wiskott et al. (Wiskott et al., 1997), an elastic graph-matching algorithm is used with a neural network for face recognition. Faces are stored as flexible graphs or grids with characteristic visual features (Gabor features) attached to the nodes of the graph (labeled graphs). The Gabor features are based on the wavelet transform, and have been shown to provide a robust information coding for object recognition (invariance against intensity or contrast changes). Furthermore, Gabor-features are less affected by pose, size and facial expression than raw grey level features. For image matching against a stored graph, the graph location in the image is optimized. It has been shown that Elastic Bunch Graph Matching (EBGM) can successfully recognize faces from facial line drawings. The efficiency of the Gabor-wavelets in recognizing line drawings is due to the fact that line drawings have dominant orientations of bars and step edges, and the Gabor-code is also dominated by orientation features. Gender classifications experiments performed on line drawings resulted in a correct-decision rate of better than 90%.
Perronnin & Dugelay (Perronnin & Dugelay, 2003) proposed a further deformable model, whose philosophy is similar to the EBGM. They introduced a novel probabilistic deformable model of face mapping, based on a bi-dimensional extension of the 1D-HMM (Hidden Markov Model). Given a template face F_T, a query face F_Q and a deformable model M, the proposed method try to maximize the likelihood $P(F_T|F_Q,M)$. There are two main differences between this method and the original EGM. First of all the HMM is extended to the 2D case to estimate $P(F_T|F_Q,M)$, automatically training all the parameters of M, so taking into account for the elastic properties of the different parts of the face. Secondly, the model M is shared among all faces, so the approach works well also when little enrolment data is available.
In Ref. (Lam & Yan, 1998), an analytic-to-holistic approach is introduced for identification of faces at different perspective variations. The ORL-database is used in the experiments. Only

one upright frontal face is selected for each of 40 individuals. Among the rest of the faces, they selected 160 images as a testing set. About half of the faces are upright and have a small rotation on the y-axis. The other half show different amounts of perspective variations. Fifteen feature points are located on a face. A head model is proposed, and the rotation of the face can be estimated using geometrical measurements. The positions of the feature points are adjusted so that their corresponding positions for the frontal view are approximated. A similarity transform is then used to compare the feature points with pre-stored features. In addition to that, eyes, nose and mouth are correlated with corresponding patterns in a database. Under different perspective variations, the overall recognition rates are over 84% and 96% for the first and the first three likely matched faces, respectively.
In Ref. (Li & Lu, 1999), a classification method, called the Nearest Feature Line (NFL), is proposed for face recognition. The line passing through two feature points in the eigenspace of the same class is used to generalise any two featurepoints of the same class. The derived FL can capture more variations of face images than the original points. A nearest distance-based classifier is used. The nearest feature line method achieved an error rate of 3.125%, and the authors claim that it is the lowest reported rate for the ORL database. The authors expect this improvement to be due to the feature lines' ability to expand the representational capacity of available feature points, and to account for new conditions not represented by original prototype face images. The error rate of the proposed method is 43.7–65.4% of that of the standard Eigenface method.

2.2 Statistics-based Face Recognition Algorithms

The Eigenface approach described by Turk and Pentland (Turk & Pentland, 1991) is one of the most popular approaches for face recognition. The principal component analysis is applied on the training set of faces. The Eigenface approach assumes that the set of all possible face images occupies a low-dimensional subspace, derived from the original high-dimensional input image space. The Eigenface space is an approximation of face patterns in the training set using data clusters and their principal components. An unknown face is classified if its distance to the clusters is below a certain threshold, using an appropriate classifier. Turk and Pentland (Turk & Pentland, 1991) reported a correct recognition rate of 95% in the case of the FERRET database, containing about 3000 different faces. The tested face images seem to be taken with little variation in viewpoint and lighting, although with significant variation in facial expression. The major drawback of the Eigenface approach is that the scatter being maximised is due not only to the 'between-class scatter' that is useful for classification, but also to the 'within-class scatter' that, for classification purposes, is unwanted information.
Many other researchers have implemented the Eigenface approach for comparison purposes. Belhumeur et al. (Belhumeur et al., 1997) used the Fisherface method to project face images into a three dimensional linear subspace. The projection is based on Fisher's Linear Discriminant in order to maximise the 'between-class scatter' while minimising the 'within-class scatter'. This approach is proved to be more efficient than the Eigenface approach, especially in the case of variable illumination. The experiments were performed on only 150 faces from 15 subjects selected from the ORL database. The results show that the Eigenface approach is quite robust when dealing with glasses and facial expressions, but sensitive to scale, pose and illumination. The correct recognition rate achieved is 95% for only 150 images, selected from the 400 images of the ORL database.

The LDA (Linear Discriminant Analysis) (Lu et al., 2003; Martinez and Kak, 2001) has been proposed as a better alternative to the PCA. It expressly provides discrimination among the classes, while the PCA deals with the input data in their entirety, without paying any attention for the underlying structure. Indeed the main aim of the LDA consists in finding a base of vectors providing the best discrimination among the classes, trying to maximize the between-class differences, minimizing the within-class ones.

3. Self-Organizing Maps (SOM)

In this section, we will give a brief introduction on Self-Organizing Map (SOM) (Kohonen, 1985). SOM has been proposed by Kohonen in the early eighties (Kohonen, 1985). Since that time, it has been used most widely for data analysis in some areas such as economics physics, chemistry or medical applications. As a general purpose clustering tool with topology preserved from input data, Self-Organizing Map (SOM) has been widely utilized in various areas now (Kohonen, 1996).

The SOM provides an orderly mapping of an input high-dimensional space $\Re^n$ in much lower dimensional spaces, usually one or two dimensions. As it compresses information while preserving the most important topological and metric relationships of the primary data items, it can be thought to produce some kind of abstractions of information. So it can be utilized in a number of ways in complex tasks such as pattern classification, process analysis, machine perception, control, and communication.

The Kohonen neural network consists of two layers; the first one (input layer) is connected to each vector of the dataset, the second one (output layer) forms a two-dimensional array of nodes. In the output layer, the units of the grid (virtual sites) give a representation of the distribution of the sample units in an ordered way. For learning, only input units are used, no expected output data is given to the system; we are referring to unsupervised learning. The learning steps are well known (Kohonen, 1996) but will be described in some detail.

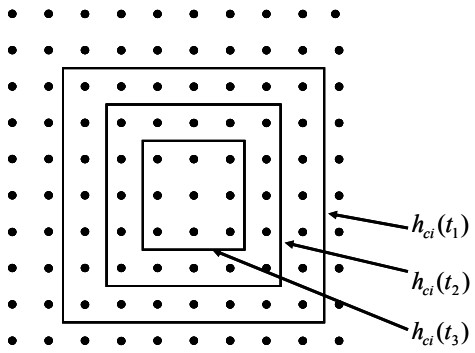


Fig. 1. An example of a two-dimensional SOM

An example of a two-dimensional SOM is show in Figure 1. With each *i*, a reference vector in the input space,

$$W_i = [w_{i1}, w_{i2}, ..., w_{in},]^T \in \Re^n,$$

is assigned to each node in the SOM. During training, each input x, is compared to all of the W_i, obtaining the location of the close match

$$|x - W_c| = \min|x - W_i| \tag{1}$$

The input point is mapped to this location in the SOM. Nodes in the SOM are updated according to:

$$W_i(t+1) = W_i(t) + h_{ci}(t)[x(t) - W_i(t)] \tag{2}$$

Where t is the time during learning and $h_{ci}(t)$ is the neighbourhood function, a smoothing kernel which is maximum at W_c. Usually,

$$h_{ci}(t) = h(\| r_c - r_i \|, t), \tag{3}$$

where r_c and r_i represent the location of the nodes in the SOM output space. r_c is the node with the closest weight vector to the input sample and r_i ranges over all nodes. $h_{ci}(t)$ approaches 0 as $\| r_c - r_i \|$ increases and also as t approaches ∞. A widely applied neighbourhood function is:

$$h_{ci} = \alpha(t) \exp(-\frac{\| r_c - r_i \|^2}{2\sigma^2(t)}) \tag{4}$$

where $\alpha(t)$ is a scalar valued learning rate and $\sigma(t)$ defines the width of the kernel. They are generally both monotonically decreasing with time. The use of the neighborhood function means that nodes which are topographically close in the SOM structure activate each other to learn something from the same input x. A relaxation or smoothing effect results which leads to a global ordering of the map. Note that $\sigma(t)$ should not be reduced too far as the map will lose its topographical order if neighboring nodes are not updated along with the closest node. The SOM can be considered a non-linear projection of the probability density, $p(x)$ (Kohonen, 1995).

4. Face Recognition Using SOM

In this section, we presents in details on various face recognition applications using SOM, including analysis and discussions of their advantages and demerits.

4.1 Overview

Figure 2 shows a general procedure of face recognition algorithm using Self-Organizing Map (SOM). For face recognition algorithm, Self-Organizing Map (SOM) usually fills the role of dimension reduction and feature extraction.

Self-Organizing maps (SOMs) (Kohonen, 1985; Kohonen, 1996) have been successfully used as a way of dimensionality reduction and feature selection for face space representations (Lawrence et al., 1997; Tan et al., 2005; Kumar et al., 2008).

In Lawrence et al. (Lawrence et al., 1997), both a convolutional neural network and a self-organising feature map classifier were used for invariant face recognition. This system was tested on the ORL database, and resulted in a correct recognition rate of 96.2% for the case of a training set, including five faces per person and a test set including five faces per person.

Neagoe (Neagoe & Ropot, 2002) present a new neural classification model called Concurrent Self-Organizing Maps (CSOM), representing a winner-takes-all collection of small SOM networks. Each SOM of the system is trained individually to provide best results for one class only. They obtained a recognition score of 91% with CSOM (40 small linear SOMs) on the ORL database.

Tan (Tan et al., 2005) proposed a Self-Organizing Map (SOM) based algorithm to solve the one training sample face recognition problem. As stated in Ref. (Tan et al., 2005), the SOM algorithm can extract local facial features even with a single image sample due to its unsupervised, nonparametric nature, resulting in a lower recognition error rate compared to PCA.

Kumar et al. (Kumar et al., 2005) integrated the two techniques for dimensionality reduction and feature extraction and to see the performance when the two are combined. Simulation results show that, though, the individual techniques SOM and PCA itself give excellent performance but the combination of these two can also be utilized for face recognition. The advantage in combining the two techniques is that the reduction in data is higher but at the cost of recognition rate.

Oravec et al. (Oravec & Pavloicova, 2007) present an original method of feature extraction from image data using MLP (multilayer perceptron) and PCA (principal component analysis). This method is used in human face recognition system and results are compared to face recognition system using PCA directly, to a system with direct classification of input images by MLP and RBF (radial basis function) networks, and to a system using MLP as a feature extractor and MLP and RBF networks in the role of classifier. Also a two stage method for face recognition is presented, in which Kohonen self-organizing map is used as a feature extractor. This method uses feature extraction method from image data, which is based on vector quantization (VQ) of images using Kohonen self-organizing map for codebook design. The indexes used for image transmission are used to recognize faces.

Aly et al. (Aly et al., 2008) present an appearance-based method for face recognition and evaluate its robustness against illumination changes. Self-organizing map (SOM) is utilized to transform the high dimensional face image into low dimensional topological space. However, the original learning algorithm of SOM uses Euclidean distance to measure similarity between input and codebook images, which is very sensitive to illumination changes. In Ref. (Aly et al., 2008) , they present Mahalanobis SOM, which uses Mahalanobis distance instead of the original Euclidean distance. The effectiveness of the proposed method is demonstrated by conducting some experiments on Yale B and CMU-PIE face databases.

Lefebvre & Garcia (Lefebvre & Garcia, 2008) present a method aiming at quantifying the visual similarity between an image and a class model. This kind of problem is recurrent in many applications such as object recognition, image classification, etc. In Ref. (Lefebvre & Garcia, 2008), they proposed to label a Self-Organizing Map (SOM) to measure image

similarity. To manage this goal, they feed local signatures associated to the regions of interest into the neural network. At the end of the learning step, each neural unit is tuned to a particular local signature prototype. During the labeling process, each image signature presented to the network generates an activity vote for its referent neuron. Facial recognition is then performed by a probabilistic decision rule. This scheme offers very promising results for face identification dealing with illumination variation and facial poses and expressions.

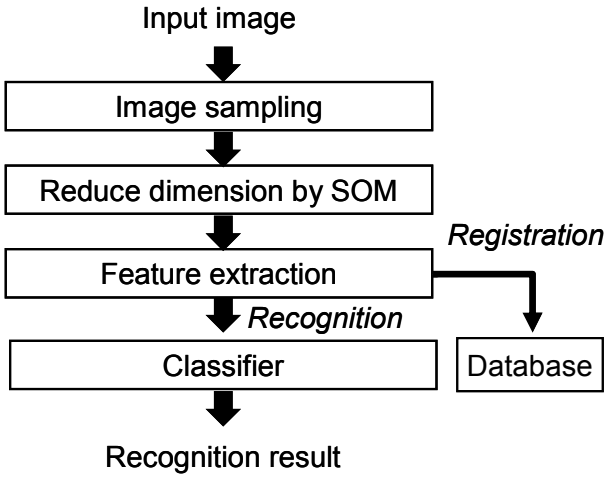


Fig. 2. General procedure of face recognition algorithm using Self-Organizing Map (SOM)

4.2 Some Typical Face Recognition Approaches Using SOMs

4.2.1 Hybrid Neural-Network method using SOM and CNN for Face Recognition

Lawrence et al. (Lawrence et al. 1997) proposed a hybrid neural-network method which combines local image sampling, a self-organizing map (SOM) neural network, and a convolutional neural network (CNN).

In Ref. (Lawrence et al. 1997), both a convolutional neural network and a self-organizing feature map classifier were used for invariant face recognition. The system was tested on the ORL database, and resulted in a correct recognition rate of 96.2% for the case of a training set, including five faces per person and a test set including five faces per person. On the contrarily, testing the Eigenface method resulted in 89.5% correct recognition rate.

The overview of this method is described as follows.

1. In their system, firstly, a window is stepped over the image and a vector is created from a local window on the image using the intensity variation given by the difference between the centre pixel and all other pixels within the square window. For the images in the training set, a fixed size window (e.g. 5 x 5) is stepped over the entire image and local image samples are extracted at each step. At each step the window is moved by 4 pixels.

2. The Self-Organizing Map (SOM) provides a quantization of the image samples into a topological space where inputs that are nearby in the original space are also nearby in the output space, thereby providing dimensionality reduction and invariance to minor changes in the image sample. A Self-Organizing Map (SOM) (e.g. with three dimensions and five nodes per dimension, $5^3 = 125$ total nodes) is trained on the vectors from the previous stage. The SOM quantizes the 25-dimensional input vectors into 125 topologically ordered values. The three dimensions of the SOM can be thought of as three features.

The SOM is used primarily as a dimensionality reduction technique and it is therefore of interest to compare the SOM with a more traditional technique. Hence, experiments were performed with the SOM replaced by the Karhunen-Lo`eve transform. In this case, the KL transform projects the vectors in the 25-dimensional space into a 3-dimensional space.

3. The same window as in the first step is stepped over all of the images in the training and test sets. The local image samples are passed through the SOM at each step, thereby creating new training and test sets in the output space of the self-organizing map. (Each input image is now represented by 3 maps, each of which corresponds to a dimension in the SOM. The size of these maps is equal to the size of the input image (92x112) divided by the step size (for a step size of 4, the maps are 23x28).
4. The convolutional neural network provides for partial invariance to translation, rotation, scale, and deformation. The convolutional network extracts successively larger features in a hierarchical set of layers, which is trained on the newly created training set. Training a standard MLP was also investigated for comparison.

The main objective of this algorithm is how to improve the robustness of recognition system using a five-layered convolutional neural network (CNN). So it can not provide a general representation for one sample per person problem.

4.2.2 SOM and Soft k-NN Ensemble for Single Training Image per Person

Towards the one training sample face recognition problem, Tan proposed a similar Self-Organizing Map (SOM) based algorithm (Tan et al., 2005) to solve it. Based on the work of Martinez's (Martinez, 2002), which used a local probabilistic method to recognize partially occluded and expression variant face from a single sample per class, Tan extended it by proposing an alternative way of representing the face subspace with SOMs instead of a mixture of Gaussians. As stated in Ref. (Tan et al., 2005), the SOM algorithm can extract local facial features even with a single image sample due to its unsupervised, nonparametric nature, resulting in a lower recognition error rate compared to PCA.

The overview of this method is described as follows.

1. Localizing the face image

   The original face image is firstly divided into M nonoverlapping subblocks with equal size. Then the M low dimensional local feature vectors (LFVs) are obtained by concatenating the pixels of each subblock.
2. SOM projection

   A SOM network is then trained using all the obtained subblocks from all the available training images irrespective of classes. After the SOM map has been trained, each subblock R_i of the same face image I are mapped to its corresponding best matching units (BMUs) by a nearest neighbor strategy, whose location in the 2D SOM topological space is denoted as a location vector $l_i = \{x_i, y_i\}$. All the location vectors from the same

face can be grouped as a set, which is called SOM-face. The merit of SOM-face is that even when the sample size is too small to faithfully represent the underlying distribution, the SOM algorithm can still extract all the significant information of local facial features due to the algorithm's unsupervised and nonparametric characteristic, while eliminating possible faults like noise, outliers, or missing values. Therefore, the compact and robust representation of the subspace can be reliably learned. In Ref. (Tan et al., 2005), based on the localization of the training images, two strategies of learning the SOM topological space are proposed, namely to train a single SOM map for all the samples and to train a separate SOM map for each class, respectively.

3. Identifying face based on SOM-face
   A soft nearest neighbor (soft -NN) ensemble decision method, which can effectively exploit the outputs of the SOM topological space and can avoid the losing of information, is used to identify the unlabeled subjects.

This algorithm was tested on the FERET database (Phillips et al., 2000), which a single SOM map was trained using all 1196 images in the gallery set. And the recognition rate of the system on 1195 probe images is measured, resulted in a correct recognition rate which outperformed the standard Eigenface technique (Turk & Pentland, 1991) and 2-DPCA (Yang et al., 2004) by 10%–15%.

However, the performance of these approaches was evaluated using frontal face patterns where only small variations are considered. If more complex variations such as aging-, illumination- and pose-variations are included, the recognition performance may be still in question (Wang et al., 2006).

4.3 Discussions

The Self-organizing Map (SOM) provides a quantization of the image samples into a topology preserving space where inputs located nearby in the original space also appear nearby in the output space. The SOM achieves dimensionality reduction and provides partial invariance to translation, rotation, scale, and deformation in the image sample.

However, when the number of people increases, the computation expenses become more demanding. In general, neural network approaches encounter problems when the number of classes (i.e., individuals) increases. Moreover, they are not suitable for a single model image recognition task because multiple model images per person are necessary in order to train the systems for optimal parameter settings (Kong et al, 2005). Tan et al. (Tan et al., 2005) have proposed a SOM-face representation to resolve the one training sample face recognition problem. Fusion of multiple neural networks classifiers improved the overall performance of face recognition (Gutta et al., 2000; Lawrence et al., 1997; Tan et al., 2005; Kumar et al., 2008).

5. Conclusions

In this chapter, various face recognition approaches using Self-Organizing Map are reviewed. The Self-Organizing Map (SOM) provides an orderly mapping of an input high-dimensional space in much lower dimensional spaces, so it can play the role of dimension reduction and feature extraction for face recognition algorithm. Furthermore, because it can provides partial invariance to translation, rotation, scale, and deformation in the image

sample, combined with other neural networks methods, more and more face recognition algorithm using SOM will be studied in the future.

6. References

Abate, A.F.; Nappi, M.; Riccio, D. & Sabatino, G. (2007). 2D and 3D Face Recognition: A Survey, *Pattern Recognition Letters*, Vol. 28, pp. 1885-1906

Aly, S. ; Tsuruta, N. & Taniguchi, R. (2008). Face Recognition under Varying Illumination Using Mahalanobis Self-organizing Map, *Artificial Life and Robotics*, Vol. 13, No. 1, pp. 298-301

AT&T Laboratories Cambridge, The database of faces, at http://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html

Bartlett, M. S. ; Movellan, J. R. & Sejnowski, T. J. (2002). Face Recognition by Independent Component Analysis, *IEEE Trans. on Neural Networks*, Vol. 13, No. 6, pp. 1450-1464

Belhumeur, P.; Hespanha, J. & Kriegman, D. (1997). Eigenfaces vs. Fisherfaces: Recognition Using Class Specific Linear Projection, *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Vol. 19, No.7, pp. 711–720

Ben, A. J. & Nandy, D. (1998). A Volumetric/iconic Frequency Domain Representation for Objects with Application for Pose Invariant Face Recognition, *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Vol. 20, No. 5, pp. 449–457

Bowyer, K.W. (2004). Face Recognition Technology and the Security Versus Privacy Tradeoff, *IEEE Technology and Society*, pp. 9-20

Brunelli, R. & Poggio, T. (1993). Face Recognition: Features Versus Templates, *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Vol. 15, No. 10, pp. 1042-1052

Chellappa, R. ; Wilson, C. L. & Sirohey, S. (1995). Human and Machine Recognition of Faces: A Survey, *Proc. IEEE*, Vol. 83, No. 5, pp. 705-740

Filippone, M.; Camastra, F.; Masulli, F. & Rovetta, S. (2008). A Survey of Kernel and Spectral Methods for Clustering, *Pattern Recognition*, Vol. 41, pp. 176 -190

Goudail, F. ; Lange, E. ; Iwamoto, T.; Kyuma, K. & Otsu, N. (1996). Face Recognition System Using Local Autocorrelations and Multiscale Integration, *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Vol. 18, No. 10, pp. 1024-1028

Gutta, S. ; Huang, J. R. J. ; Jonathon, P. & Wechsler, H. (2000). Mixture of Experts for Classification of Gender, Ethnic Origin, and Pose of Human Faces, *IEEE Trans. on Neural Network*, Vol. 11, Issue. 4, pp. 948–960

Karungaru, S.G.; Fukumi, M. & Akamatsu, N. (2005). Face Recognition in Colour Images Using Neural Networks and Genetic Algorithms, *International Journal of Computational Intelligence and Applications*, Vol. 5, No. 1, pp. 55-67

Kohonen, T. (1985). *Self-Organizing Maps*, Springer-Verlag, Berlin

Kohonen, T. (1996). Engineering Applications of the Self-Organizing Map, *Proceedings of the IEEE*, Vol. 84, No. 10, pp. 1358-1384

Kong, S. G.; Heo, J.; Abidi, B. R.; Paik, J. & Abidi, M. A. (2005). Recent Advances in Visual and Infrared Face Recognition - A Review, *Computer Vision and Image Understanding*, Vol. 97, pp. 103-135

Kotani, K. ; Chen, Q. & Ohmi, T. (2002). Face Recognition Using Vector Quantization Histogram Method, *IEEE 2002 International Conference on Image Processing*, II-105-108

Kumar, D.; Rai, C.S. & Kumar, S. (2005). Face Recognition Using Self-Organizing Map and Principal Component Analysis, *International Conference on Neural Networks and Brain 2005 (ICNN&B '05)*, Vol. 3, pp. 1469-1473

Lam, K.-M. & Yan, H. (1998). An Analytic-to-holistic Approach for Face Recognition Based on a Single Frontal View, *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Vol. 20, No. 7, pp. 673-686

Lampinen, J. & Oja, E. (1995). Distortion Tolerant Pattern Recognition Based on Self-Organizing Feature Extraction, *IEEE Trans. on Neural. Network*, Vol. 6, pp. 539-547

Lawrence, S.; Giles, C. L.; Tsoi, A. C. & Back, A. D. (1997). Face Recognition: A Convolutional Neural Network Approach, *IEEE Trans. on Neural Networks*, Vol. 8, No.1, pp.98-113.

Lefebvre, G & Garcia, C. (2008). A Probabilistic Self-Organizing Map for Facial Recognition, *19th International Conference on Pattern Recognition*, Florida, USA

Li, S. Z. & Lu, J. (1999). Face Recognition Using the Nearest Feature Line Method, *IEEE Trans. on Neural Networks*, Vol. 10, No. 2, pp. 439–443

Li, S. Z. & Jain, A. K. (2005). *Handbook of Face Recognition*, Springer, New York

Lin, S. H. ; Kung, S. Y. & Lin, L. J. (1997). Face Recognition/detection by Probabilistic Decision-based Neural Network, *IEEE Trans. on Neural Networks*, Vol. 8, No. 1, 114–132

Martinez, A. M. (2002). Recognizing Imprecisely Localized, Partially Occluded, and Expression Variant Faces from a Single Sample per Class, *IEEE Trans. on Pattern Analysis Machine Intelligence*, Vol. 25, No. 6, pp. 748-763

Moghaddam, B. & Pentland, A. (1997). Probabilistic Visual Learning for Object Representation, *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Vol. 19, No. 7, pp. 696-710

Neagoe, V. E. & Ropot, A. D. (2002). Concurrent Self-Organizing Maps for Pattern Classification, *Proceedings of the First IEEE International Conference on Cognitive Informatics (ICCI02)*

Oravec, M. & Pavloicova, J. (2007). Face Recognition Methods Based on Feedforward Neural Networks, Principal Component Analysis and Self-Organizing Map, *Radioengineering*, Vol. 16, No. 1, pp. 51-57

Penev, P. S. & Atick, J. J. (1996). Local Feature Analysis: A General Statistical Theory for Object Representation, *Network: Computation in Neural Systems*, Vol. 7, No. 3, pp. 477-500

Perronnin, F. & Dugelay, J.-L. (2003). An Introduction to Biometrics and Face Recognition, *Proc. IMAGE'2003: Learning, Understanding, Information Retrieval, Medical*, Cagliari, Italy

Phillips, P. J.; Moon, H.; Rizvi, S. A. & Rauss, P. J. (2000). The FERET Evaluation Methodology for Face-Recognition Algorithms, *IEEE Trans. on Pattern Recognition and Machine Intelligence*, Vol. 22, No. 10, October 2000, pp. 1090-1104

Tan, X.; Chen, S.; Zhou, Z. & Zhang, F. (2005). Recognizing Partially Occluded, Expression Variant Faces from Single Training Image per Person with SOM and Soft k-NN Ensemble, *IEEE Trans. on Neural Networks*, Vol. 16, No. 4, pp. 875-886

Tan, X.; Chen, S.; Zhou, Z. & Zhang, F. (2006). Face Recognition from A Single Image per Person: A Survey, *Pattern Recognition*, Vol. 39, pp. 1725-1745

Tolba, A. S. & Abu-Rezq, A. N. (2000). Combined Classifiers for Invariant Face Recognition, *Pattern Analysis & Applications*, Vol. 3, pp. 289-302

Turk, M. & Pentland, A. (1991). Eigenfaces for Recognition, *Journal of Cognitive Neuroscience*, Vol. 3, No. 1, pp. 71-86

Wang, J.; Plataniotis, K. N.; Lu, J. & Venetsanopoulos, A. N. (2006). On Solving the Face Recognition Problem with One Training Sample per Subject, *Pattern Recognition*, Vol. 39, pp. 1746 -1762

Wiskott, L.; Fellous J. M.; Kruger, N. & Malsburg, C. von der. (1997). Face Recognition by Elastic Bunch Graph Matching, *IEEE Trans. Pattern Analysis and Machine Intelligence*, Vol. 19, No. 7, pp. 775-779

Yang, J.; Zhang, D.; Frangi, A. F. & Yang, J. (2004). Two-dimensional PCA: A New Approach to Appearance-based Face Representation and Recognition, *IEEE Trans. on Pattern Analysis Machine Intelligence*, Vol. 26, No. 1, pp. 131–137

Zhang, D. ; Peng, H. ; Zhou, J. & Pal S. K. (2002). A Novel Face Recognition System Using Hybrid Neural and Dual Eigenspaces Methods, *IEEE Trans. on System Man Cybernetics-Part A*, Vol. 32, No. 6, pp. 787-793

Zhao, W. (2000). Discriminant Component Analysis for Face Recognition, *Proceeding of International Conference on Pattern Recognition*, Track 2, pp. 822-825

Zhao, W. ; Chellappa, R. ; Rosenfeld, A. & Phillips, P.J. (2003). Face Recognition: A Literature Survey, *ACM Computing Surveys*, Vol. 35, Issue 4, pp. 399-458

18

Generation of Emotional Feature Space for Facial Expression Recognition Using Self-Mapping

Masaki Ishii[1] and Makoto Nishida[2]
[1]Akita Prefectural University, [2]Akita University
[1,2]Japan

1. Introduction

The study of facial expression recognition for the purpose of man-machine emotional communication is attracting attention lately (Akamatsu, 2002a; Akamatsu, 2002b; Akamatsu, 2002c; Akamatsu, 2003; Fasel & Luettin, 2003; Pantic & Rothkrantz, 2000; Tian et al., 2001).
The shape (static diversity) and motion (dynamic diversity) of facial components such as the eyebrows, eyes, nose, and mouth manifest expression. Considering facial expression from a viewpoint of static diversity, because facial configurations differ for every person, it is presumed that a facial expression pattern appearing on a face when facial expression is manifested includes subject-specific features. In addition, from the viewpoint of dynamic diversity, because the dynamic change of facial expression originates in a subject-specific facial expression pattern, it is presumed that the displacement vector of facial components has subject-specific features.
On the other hand, although a facial expression pattern appearing on a face by emotion is peculiar to an individual, the internal emotion that humans express on the face and the emotion that humans recognize from the facial expression are considered to be person-independent and universal. For example, one expresses the common emotion of happiness using various facial expressions, whereas another person might recognize the common emotion of happiness from these various facial expressions. Pantic et al. argue that a natural facial expression always includes various emotions, and that a pure facial expression rarely appears (Pantic & Rothkrantz, 2000). Furthermore, they suggest that it is not realistic to classify all facial expressions appearing on a face into one of the six basic emotion categories: anger, sadness, disgust, happiness, surprise and fear, and that quantitative classification into many emotion categories must be performed instead.
Precedent works on the quantification of emotion recognized from facial expression are found in the field of psychology. Especially, the mental space model of Russell et al. is well known: each facial expression is arranged on a space centering on ”pleasantness” and ”arousal”, particularly addressing the semantic antithetical nature of emotion (Russell & Bullock, 1985). Russell et al. discovered that facial expression stimuli can be conceptualized as arranged in a circle in the mental space described above (The Circumplex

Model). Yamada found a significant correlation between "slantedness" and "curvedness/openness" of facial components and "pleasantness" and "arousal" on the mental space (Yamada, 2000). This observation underscores the importance of clarifying a relationship of correspondence between the change of facial components accompanying emotional expression (physical parameters) and recognized emotion (psychological parameters).

We propose the following two subjects for recognizing emotion from facial expression.

The first subject concerns a facial expression pattern as a physical parameter. Expression carries personality, and facial expression patterns, as physical parameters, differ among people. For those reasons, the classification problem of facial expression is fundamentally a problem with an unknown number of categories. Therefore, it is an important subject how subject-specific facial expression categories are extracted using a common and person-independent technique.

The second subject relates to emotion as a psychological parameter. Although a facial expression pattern is peculiar to an individual, emotion as a psychological parameter is person-independent and universal. Moreover, the grade of the emotion that is recognized changes according to that of physical change of a facial expression pattern. It is therefore considered a key subject to match the amount of physical change of a subject-specific facial expression pattern with that of psychological change corresponding to that extent, to estimate the grade of emotion.

This chapter presents a generation method of a subject-specific emotional feature space using the Self-Organizing Maps (SOM) (Kohonen, 1995) and the Counter Propagation Networks (CPN) (Nielsen, 1987). The feature space expresses the correspondence relationship between the change of facial expression pattern and the strength of emotion on the two-dimensional space centering on "pleasantness" and "arousal".

Specifically, the proposed method first hierarchically classified facial expression images using a SOM, and extracted subject-specific facial expression categories. Next, data expansion of facial expression patterns based on the similarity and continuity of each facial expression category was performed using CPN. Then a subject-specific emotion feature space was generated; the space matches a physical and psychological parameter by inputting the coordinate value on the Circumplex model of Russell as a teaching signal of CPN.

Experimental results suggested that our method was useful to estimate strength and mixture level of six basic emotions.

2. Algorithms of SOM and CPN

2.1 Self-Organizing Maps (SOM)

The SOM is a learning algorithm that models the self-organizing and adaptive learning capabilities of a human brain (Kohonen, 1995). A SOM comprises two layers: an input layer, to which training data are supplied; and a Kohonen layer, in which self-mapping is performed by competitive learning. The learning algorithm of a SOM is described below.

1) Let $w_{i,j}(t)$ be a weight from an input layer unit i to a Kohonen layer unit j at time t. Actually, $w_{i,j}$ is initialized using random numbers.

2) Let $x_i(t)$ be input data to the input layer unit i at time t; calculate the Euclidean distance d_j between $x_i(t)$ and $w_{i,j}(t)$ using (1).

$$d_j = \sqrt{\sum_{i=1}^{I} \left(x_i(t) - w_{i,j}(t)\right)^2} \tag{1}$$

3) Search for a Kohonen layer unit to minimize d_j, which is designated as a winner unit.
4) Update the weight $w_{i,j}(t)$ of a Kohonen layer unit contained in the neighborhood region of the winner unit $N_c(t)$ using (2), where $a(t)$ is a learning coefficient.

$$w_{i,j}(t+1) = w_{i,j}(t) + \alpha(t)\left(x_i(t) - w_{i,j}(t)\right) \tag{2}$$

5) Repeat processes 2)–4) up to the maximum iteration of learning.

2.2 Counter Propagation Networks (CPN)

The CPN is a learning algorithm that combines the Grossberg learning rule with the SOM (Nielsen, 1987). A CPN comprises three layers: an input layer to which training data are supplied; a Kohonen layer in which self-mapping is performed by competitive learning; and a Grossberg layer, which labels the Kohonen layer by the counter propagation of teaching signals. A CPN is useful for automatically determining the label of a Kohonen layer when a category in which training data will belong is predetermined. This labeled Kohonen layer is designated as a category map. The learning algorithm of a CPN is described below.
1) Let $w^i_{n,m}(t)$ and $w^j_{n,m}(t)$ respectively indicate weights to a Kohonen layer unit (n, m) at time t from an input layer unit i and from a Grossberg layer unit j. In fact, $w^i_{n,m}$ and $w^j_{n,m}$ are initialized using random numbers.
2) Let $x_i(t)$ be input data to the input layer unit i at time t, and calculate the Euclidean distance $d_{n,m}$ between $x_i(t)$ and $w^i_{n,m}(t)$ using (3).

$$d_{n,m} = \sqrt{\sum_{i=1}^{I} \left(x_i(t) - w^i{}_{n,m}(t)\right)^2} \tag{3}$$

3) Search for a Kohonen layer unit to minimize $d_{n,m}$, which is designated as a winner unit.
4) Update weights $w^i_{n,m}(t)$ and $w^j_{n,m}(t)$ of a Kohonen layer unit contained in the neighborhood region of the winner unit $N_c(t)$ using (4) and (5), where $a(t)$, $\beta(t)$ are learning coefficients, and $t_j(t)$ is a teaching signal to the Grossberg layer unit j.

$$w^i{}_{n,m}(t+1) = w^i{}_{n,m}(t) + \alpha(t)\left(x_i(t) - w^i{}_{n,m}(t)\right) \tag{4}$$

$$w^j{}_{n,m}(t+1) = w^j{}_{n,m}(t) + \beta(t)\left(t_j(t) - w^j{}_{n,m}(t)\right) \tag{5}$$

5) Repeat processes 2)–4) up to the maximum iteration of learning.

6) After learning is completed, compare weights $w_{n,m}^{j}$ observed from each unit of the Kohonen layer; and let the teaching signal of the Grossberg layer with the maximum value be the label of the unit.

3. Proposed Method

Figure 1 depicts the procedure used for the proposed method. The proposed method consists of following three steps.

Step1: Extraction of subject-specific facial expression categories using the SOM.
Step2: Generation of a Facial Expression Map using the CPN.
Step3: Generation of a Emotion Map using the CPN.

The proposed method is explained in detail below.

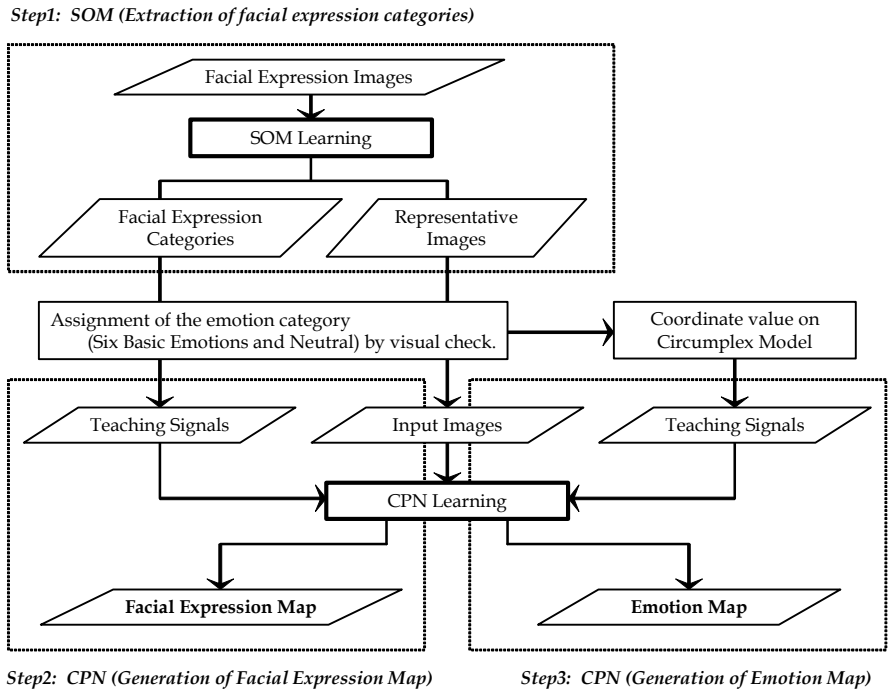


Fig. 1. Flow chart of proposal method.

3.1 Extraction of Facial Expression Category

The proposed method was used in an attempt to extract a subject-specific facial expression category hierarchically using a SOM with a narrow mapping space. A SOM is an unsupervised learning algorithm and classifies given facial expression images self-organizedly based on their topological characteristics. For that reason, it is suitable for a classification problem with an unknown number of categories. Moreover, a SOM compresses the topological information of facial expression images using a narrow mapping space and performs classification based on features that roughly divide the training data. We speculate that repeating these hierarchically renders the classified amount of change of facial expression patterns comparable; thereby, a subjectspecific facial expression category can be extracted. Figure 2 depicts the extraction procedure of a facial expression category. Details of the process are explained below.

1) Expression images described in Section 4 were used as training data. The following processing was performed for each facial expression. The number of training data is assumed as N frames.

2) Learning was conducted using a SOM with a Kohonen layer of five units and an input layer of 40 × 48 units (Fig. 2(a)), where the number of learning sessions t was set as 10,000 times. The initial value of neighborhood $N_c(t)$ was set as the forth neighbor of a winner unit. Then $N_c(t)$ was linearly reduced to the first neighbor at the maximum learning number. The initial value of learning coefficient $a(t)$ was set to 0.5, and was $a(t)$ linerly reduced to zero similarly.

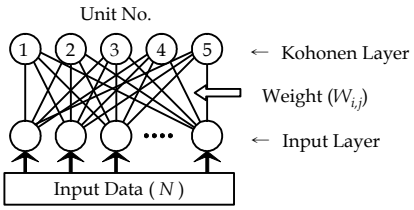
3) The weight of the Kohonen layer $W_{i,j}(0 \leq W_{i,j} \leq 1)$ was converted to a value of 0-255 after the end of learning, and a visualized images were generated (Fig. 2(b)), where $n_1 - n_5$ are the number of training data classified into each unit.

4) Five visualized images can be considered as representative vectors of the training data classified into each unit ($n_1 - n_5$). Therefore, whether a visualized image was suitable as a representative vector was judged using a threshold process. Specifically, for the upper and lower faces presented in Fig. 2(c), a correlation coefficient between a visualized image and classified training data was determined for each unit. The standard deviation of those values was computed. When the standard deviation of both regions was 0.005 or less in all five units, the visualized image was considered to represent training data and the subsequent hierarchization processing was cancelled. It was continued when any unit over 0.005 remained.

5) The correlation coefficient of weight $W_{i,j}$ between each adjacent unit in the Kohonen layer was computed. The Kohonen layer was divided into two bordering on between the units of the minimum (Fig. 2(b)).

6) The training data (N_1 and N_2) classified into both sides of the border were used as new training data; processing described above was repeated recursively. Consequently, the hierarchic structure of a SOM was generated (Fig. 2(b), 2(d)).

7) The lowermost hierarchy of the hierarchic structure was defined as a facial expression category (Fig. 2(e)). Five visualized images were defined as representative images of each category after learning completion. Then the photographer of the facial expression images performed visual confirmation to each facial expression category and conducted implication in emotion categories.



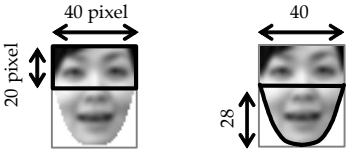
(a) Structure of SOM.

Unit No.	1	2	3	4	5
Visualized Image ($W_{i,j}$)					
Classification Result	n_1	n_2	n_3	n_4	n_5
Correlation Coefficient	0.9853	**0.9786**	0.9794	0.9866	
New Training Data	N_1		N_2		

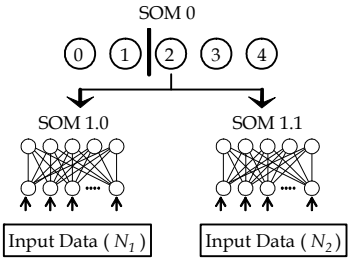
* $N = n_1 + n_2 + n_3 + n_4 + n_5$

* $N_1 = n_1 + n_2$, $N_2 = n_3 + n_4 + n_5$

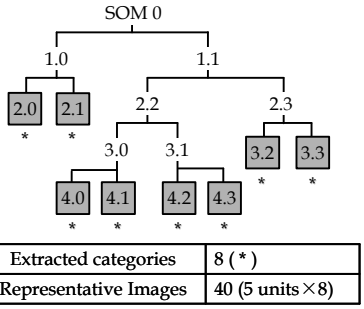
(b) Learning with SOM and setup of new training data.



(c) Target region (Upper and Lower face).



(d) Hierarchical learning with SOM.



Extracted categories	8 (*)
Representative Images	40 (5 units×8)

(e) Generation of hierarchical structure.

Fig. 2. Extraction procedure of a facial expression category.

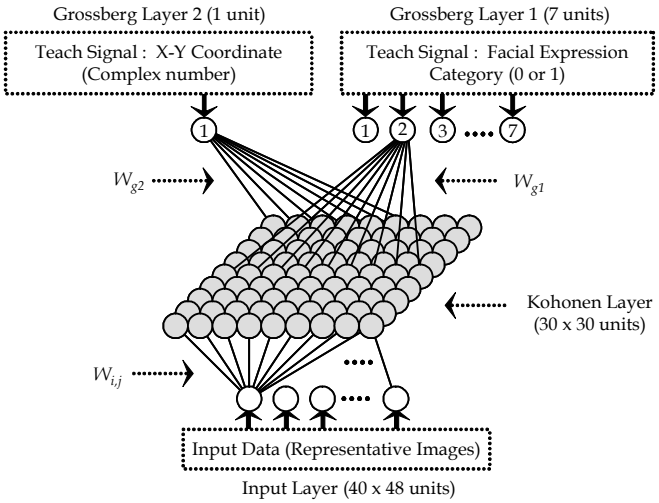
3.2 Generation of Facial Expression Map

It is considered that recognition to a natural facial expression requires generation of a facial expression pattern (mixed facial expression) that interpolates each emotion category. The proposed method used the representative image obtained in Section 3.1 as training data and carried out data expansion of facial expression patterns between each emotion category using CPN with a large mapping space. The reason for adopting CPN, a supervised learning algorithm, is that the teaching signal of training data is known by processing in Section 3.1. The mapping space of CPN has a greater number of units than the number of training data, and has a torus structure because it is presumed that a large mapping space allows CPN to perform data expansion based on the similarity and continuity of training data. Figure 3 depicts the generation procedure of facial expression map and emotion map described in Section 3.3. The details of processing are described below.

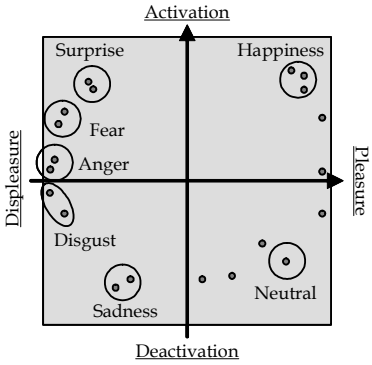
1) In fact, CPN has a structure comprising an input layer of 40 × 48 units and a two-dimensional Kohonen layer of 30 × 30 units. In addition, the Grossberg layer 1 of seven units was prepared, to which the teaching signal of six basic facial expressions and a neutral facial expression were input (Fig. 3(a)).
2) Representative images obtained in Section 3.1 were used as training data, and learning was carried out for each subject. As the teaching signal to the Grossberg layer 1, 1 was input into units that mean emotion categories of representative images, otherwise 0. The number of learning was set to 20,000 times. The initial value of the radius of neighborhood $N_c\ (t)$ was set as 1/2 of that of Kohonen layer. Then the radius was reduced linearly to the first neighbor at the maximum learning number. The initial values of training rate coefficients $a(t)$ and $\beta(t)$ were both set to 0.5. Then they were linearly reduced to zero similarly.
3) The processing described above was repeated to the maximum learning number.
4) The weights (W_{g1}) of the Grossberg layer 1 were compared for each unit of the Kohonen layer after learning completion; an emotion category of the greatest value was used as the label of the unit. A category map generated by the processing described above was defined as a subject-specific facial expression map (FEMap).

3.3 Generation of Emotion Map

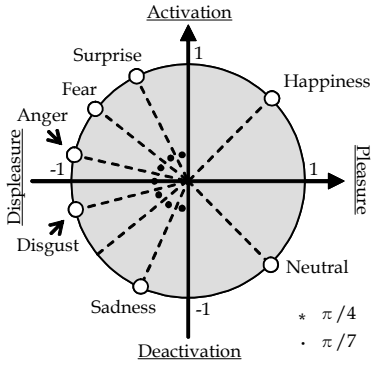
Even if the facial expression pattern appearing on a face is peculiar to an individual, the internal emotion that humans express on the face and the emotion that humans recognize from the facial expression are considered to be person-independent and universal. Therefore, it is presumed necessary to match the grade of emotion based on a common index for each subject to the grade of change of facial expression patterns extended in Section 3.2. The proposed method is centered upon the Circumplex model of Russell (Russell & Bullock, 1985) as a common index. Specifically, the coordinate values based on the Circumplex model were input as teaching signals of CPN, in parallel to processing in Section 3.2. Then generation of an emotion feature space was tried, which matches the grade of change of facial expression patterns and the grade of emotion. The details of processing are described as follows.
1) The Grossberg layer 2 of one unit that inputs the coordinate values of the Circumplex model was added to the CPN structure (Fig. 3(a)).
2) Each facial expression stimulus is arranged in a circle on a space centering on "pleasantness" and "arousal" in the Circumplex model (Fig. 3(b)). The proposed method expresses this circular space as the complex plane depicted in Fig. 3(c), and complex number based on the figure were input to the Grossberg layer 2 as teaching signals. For example, when a inputted training data is an emotion category of happiness, a teaching signal for Grossberg layer 2 is $\cos(\pi/4) + i \sin(\pi/4)$.
3) This processing was repeated to the maximum learning number.
4) Each unit of the Kohonen layer was plotted onto the complex plane after learning completion based on the values of the real and imaginary parts of the weight (W_{g2}) on Grossberg layer 2. Then this complex plane was defined as a subject-specific emotion map (EMap).



(a) Structure of CPN.



(b) Circumplex model of Russell.



(c) Complex plane expression.

Fig. 3. Generation procedure of facial expression map (FEMap) and emotion map (EMap).

4. Expression Images

Open facial expression databases are generally used in conventional studies (Pantic et al., 2005; Gross, 2005). These databases contain a few images per expression and subject. For this study, we obtained facial expression images of ourselves because the proposed method extracts person-specific facial expression categories and the representative images of each category from large quantities of data.

This chapter presents a discussion of six basic facial expressions and a neutral facial expression that four subjects manifested intentionally. Basic facial expressions were obtained as motion videos including a process in which a neutral facial expression and facial expressions were manifested five times respectively by turns for each facial expression.

Neutral facial expressions were obtained as a motion video for about 20 s. The motion videos were converted into static images (30 frame/s, 8 bit gray, 320 × 240 pixels) and used as training data. Table 1 presents the number of frames of all the subjects' training data. A region containing facial components was processed in this chapter; extraction and normalization of a face region image were performed according to the following procedures. Figure 4 shows an example of face region images after extraction and normalization.

1) A face was detected using Haar-like features (Lienhart & Maydt, 2002); a face region image normalized into a size of 80 × 96 pixels was extracted.
2) The image was processed using a median filter for noise removal. Then smoothing processing was performed after dimension reduction of the image using coarse grain processing (40 × 48 pixels).
3) A pseudo outline that is common to all the subjects was generated; the face region containing facial components was extracted.
4) Histogram linear transformation was performed for brightness value correction.

ID	An.	Sa.	Di.	Ha.	Su.	Fe.	Ne.	Total
A	454	448	533	530	473	479	605	3,522
B	284	337	323	335	325	323	599	2,526
C	493	556	592	475	455	471	599	3,641
D	574	518	457	473	459	468	605	3,554

Table 1. Number of frames of all subjects' training data (ID, Subject; An., Anger; Sa., Sadness; Di., Disgust; Ha., Happiness; Su., Surprise; Fe., Fear; Ne., Neutral).

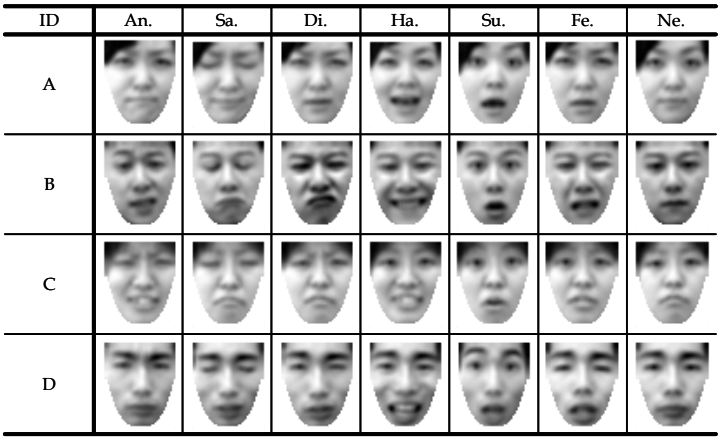


Fig. 4. Examples of facial expression images (ID, Subject; An., Anger; Sa., Sadness; Di., Disgust; Ha., Happiness; Su., Surprise; Fe., Fear; Ne., Neutral).

5. Results and Discussion

5.1 Extraction of Facial Expression Categories

Tables 2 and 3 present the number of facial expression categories and representative images for each subject extracted using the proposed method, which demonstrates that the number of facial expression categories differ for each subject. For example, for anger, 6 and 8 categories were extracted respectively from Subjects A and C, whereas only 3 and 2 categories were extracted respectively from Subjects B and D. This fact suggests that Subjects A and C have more facial expression patterns to express anger compared with Subjects B and D.

Figure 5 presents the hierarchical structure of a SOM for the happiness of Subject A. Expression categories extracted from the figure comprise eight categories of Happiness and 11 categories of neutral facial expressions, from which 95 representative images were generated. Figure 6 shows the facial expression category of happiness, in which categories of different grades of change of facial expression patterns are extracted, such as mouth wide open, mouth narrowly open, eye closed and mouth shut, eye closed and mouth open, and smile. The topological characteristic of facial expression images differs for every subject with the personality of facial expression. Because the proposed method performs hierarchical classification based on topological characteristics that are peculiar to a subject, it is presumed that the extraction result of a different facial expression category was obtained for each subject.

These results indicate that hierarchical use of SOM with a narrow mapping space is useful as a category extraction technique for facial expressions that are peculiar to an individual. It is particularly applicable when the classification problem of facial expression is considered to be a classification problem with an unknown number of categories.

( ) : Neutral

ID	An.	Sa.	Di.	Ha.	Su.	Fe.	Ne.	Total
A	6 (7)	6 (10)	8 (6)	8 (11)	4 (13)	6 (12)	0 (9)	106
B	3 (5)	4 (1)	6 (6)	6 (6)	6 (6)	9 (6)	0 (6)	70
C	8 (7)	8 (4)	1 (5)	8 (8)	5 (8)	7 (10)	0 (2)	81
D	2 (1)	4 (8)	9 (6)	8 (10)	5 (9)	8 (6)	0 (1)	77

Table 2. Number of facial expression categories extracted with proposed method.

ID	An.	Sa.	Di.	Ha.	Su.	Fe.	Ne.	Total
A	30	30	40	40	20	30	340	530
B	15	20	30	30	30	45	180	350
C	40	40	5	40	25	35	220	405
D	10	20	45	40	25	40	205	385

Table 3. Number of representative images of each category.

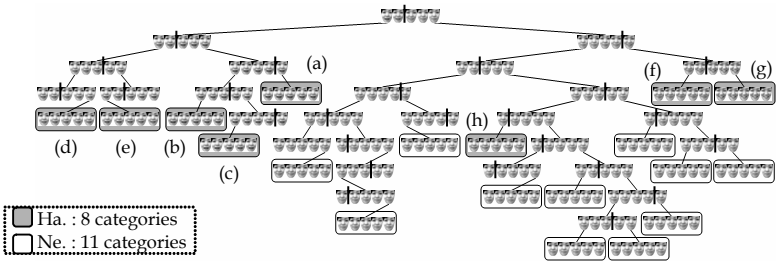


Fig. 5. Hierarchical structure of SOM (happiness of Subject A).

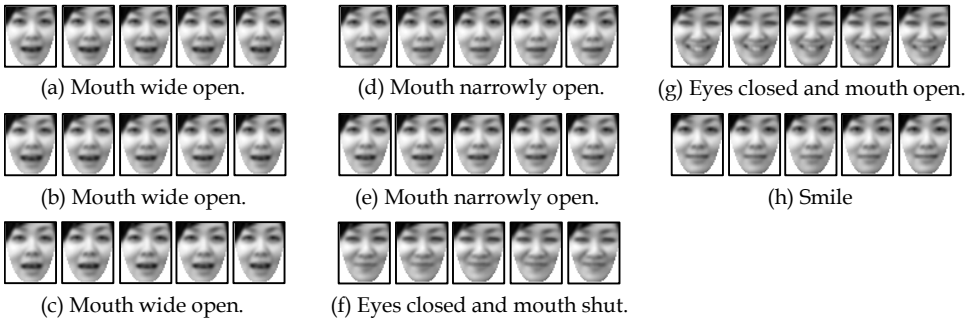


Fig. 6. Representative images (Happiness of Subject A).

5.2 Generation of FEMap and EMap

Figures 7 and 8 respectively present the FEMaps and EMaps generated using the proposed method.

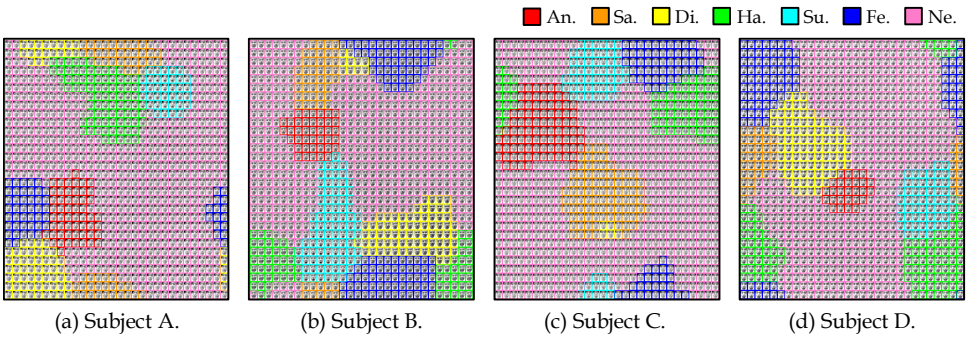
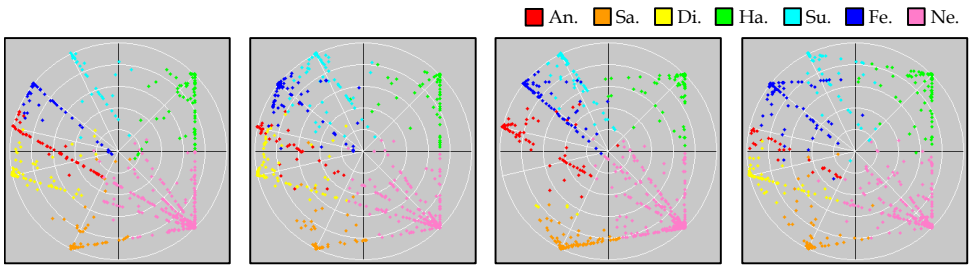


Fig. 7. Generation results of Facial Expression Map (FEMap).



(a) Subject A. (b) Subject B. (c) Subject C. (d) Subject D.

Fig. 8. Generation results of Emotion Map (EMap).

A FEMap was generated as depicted in Fig. 7, in which the adjacency and area of emotion categories differs for each subject. Table 4 shows the number of units of each emotion category on the FEMap. Table 5 provides the number of winner units when inputting the training data of Table 3 into the FEMap. For example, 252 out of 900 were winner units for Subject A. This implies that the remaining 648 units were generated as units that interpolate between training data by learning of CPN.

ID	An.	Sa.	Di.	Ha.	Su.	Fe.	Ne.	Total
A	54	56	78	82	36	57	537	900
B	40	51	84	69	88	112	456	900
C	90	87	5	81	62	78	497	900
D	27	47	89	109	55	102	471	900

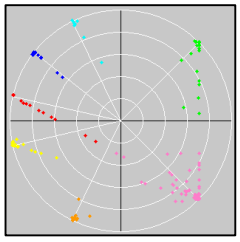
Table 4. Number of units of each emotion category on FEMap.

ID	A	B	C	D
Number of Kohonen layer units	900	900	900	900
Number of winner units (Inputs are training data)	252	183	244	231
Number of extended units	648	717	656	669

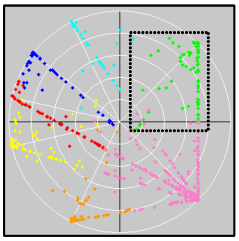
Table 5. Number of winner units when inputting training data.

On the other hand, the EMap was generated as a feature space based on a common index for each subject, as portrayed in Fig. 8. Figure 9 presents details of the EMap of Subject A, where (a) presents the plot of winner units when inputting training data, and (b) shows the plot of all units (900 units). Figure 9(a) portrays that winner units, when inputting training data, concentrate near the coordinate values of the teaching signal of CPN, based on the Circumplex model of Russell (Fig. 3(c)). On the other hand, Fig. 9(b) reveals that the extended units interpolate the result of Fig. 9(a). The region of happiness shown in Fig. 9(b) is magnified into Fig. 9(c). Because region A in Fig. 9(c) is near the coordinate value of the

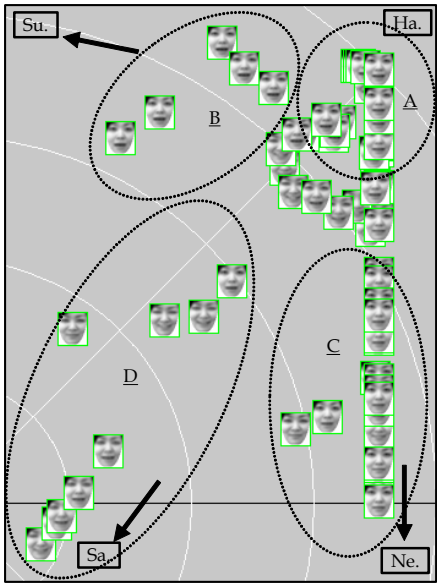
teaching signal of happiness, it is suggested that facial expressions representing pure happiness are plotted in this region. Meanwhile, because Region C shows less "arousal" than Region A, facial expressions of a transition process from happiness to neutral is plotted in this region. Similarly, facial expressions of the emotion of surprise are mixed in Region B; facial expressions of sadness are mixed in Region D.



(a) Plotted result of winner units (252 units).



(b) Plotted result of all units (900 units).



(c) Magnification of "happiness".

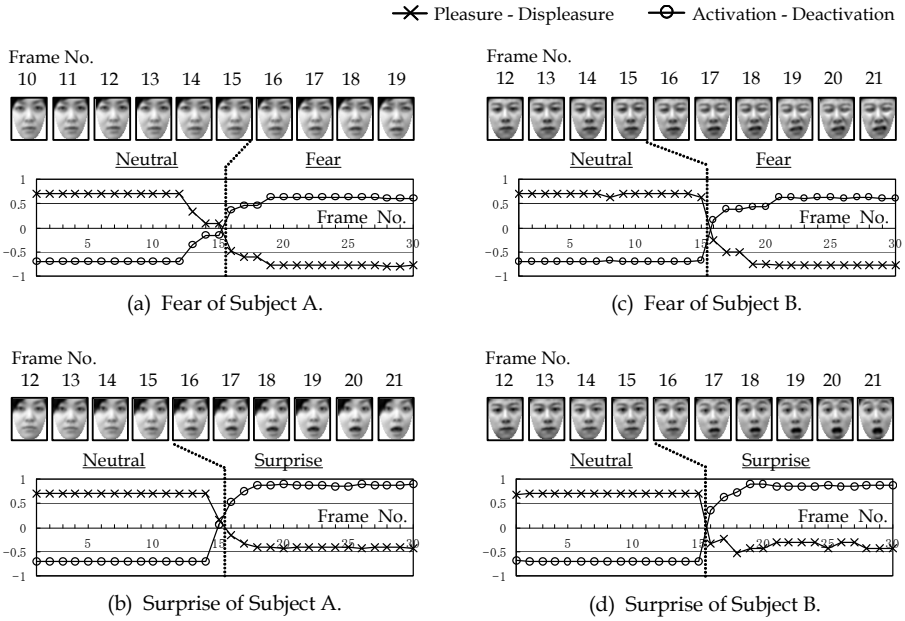
Fig. 9. Detailed analysis of EMap (Happiness of Subject A).

These results suggest that data expansion is performed based on the similarity and continuity of training data, and that more facial expression patterns such as mixed facial expressions between emotion categories can be generated in the mapping space of CPN. It is also inferred that the grade of emotion with "pleasantness" and "arousal" as indices can be matched to the grade of change of facial expression patterns on the EMap by assigning coordinate values based on the Circumplex model of Russell as a teaching signal of CPN.

6. Evaluation Experiments

An experiment was conducted that was expected to estimate the grade of emotion using time series data of newly obtained facial expressions to verify the usefulness of the proposed method. Test data for six basic facial expressions were obtained in the same conditions as those described in Section 4.

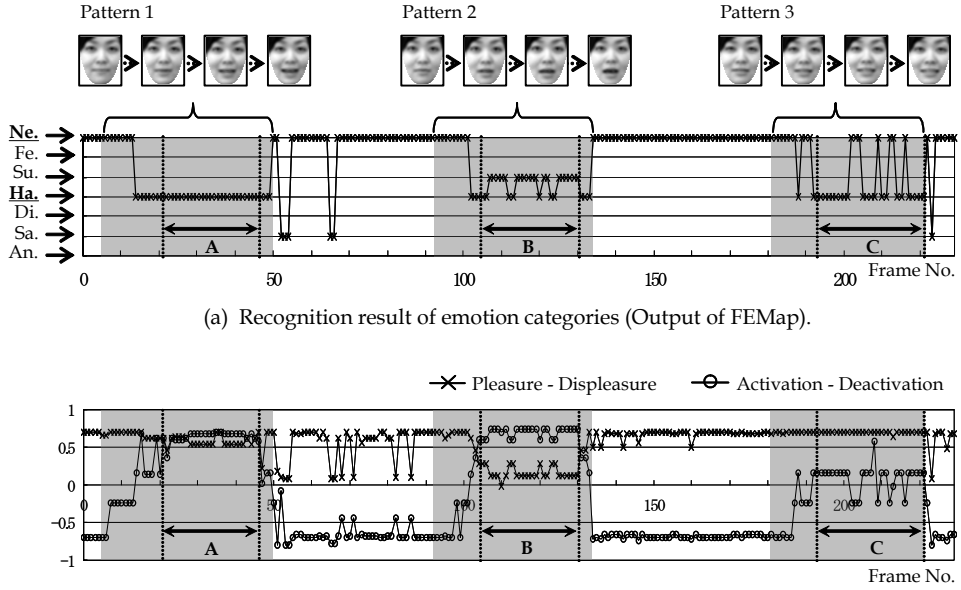
Figure 10 shows the recognition result for "fear" and "surprise" of Subject A and B, which reveals pleasantness value and arousal value gradually change with the change of facial expression pattern. Moreover, the change of pleasantness value and arousal value is similar, although facial expression patterns of two subjects are different.



(a) Fear of Subject A. (c) Fear of Subject B.

(b) Surprise of Subject A. (d) Surprise of Subject B.

Fig. 10. Recognition result for "fear" and "surprise" of Subject A and B.



(a) Recognition result of emotion categories (Output of FEMap).

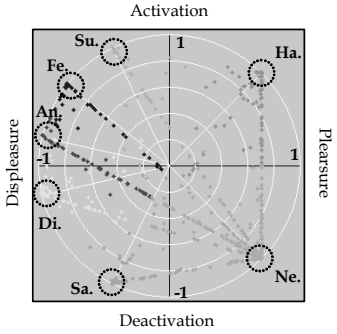
(b) Change of the grade of emotions with pleasantness and arousal (Output of EMap).

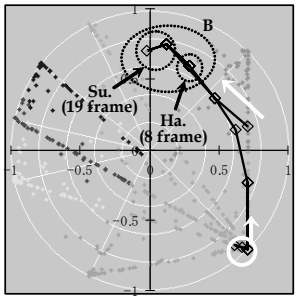
Fig. 11. Recognition result of happiness which is different facial pattern.

Figure 11 depicts the recognition result for the happiness of Subject A. The test data contain facial expressions of three patterns of happiness for which the way of manifestation differ, as shown in Fig. 11(a). That figure portrays that all the frames in Section A are recognized as happiness, where "pleasantness" and "arousal" remain high, as shown in Fig. 11(b). Section B is recognized as happiness or surprise, and "pleasantness" remains low compared to Section A; Section C is recognized as happiness or neutral, and "arousal" remains low.

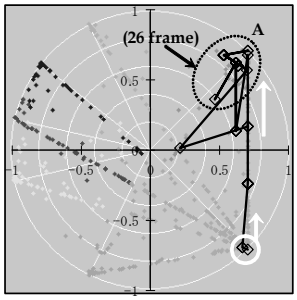
Figure 12 expresses the state transition of Patterns 1, 2, and 3 on an EMap. Regions A, B, and C in the figure mark regions in which the frames of Sections A, B, and C in Fig. 11 are plotted. Although "arousal" increases from near the coordinate value of neutral expression gradually in Patterns 1, 2 and 3, they deviate from one another when the values turn positive, as presented in Fig. 12. The state of Pattern 1 is stabilized near the coordinate value of happiness (region A). Therefore, it is surmised that these time series data manifest the typical emotion of happiness. In addition, because Pattern 2 is stable near the middle of happiness and surprise (region B), it is conjectured to represent the state of excitement and upsurge in which the emotion of happiness and surprise is mixed. Furthermore, because Pattern 3 is stable near the middle of happiness and neutral expression (region C), it is inferred that a state of satisfaction and relief is expressed, i.e. low-level manifestation of happiness.

These results demonstrate that an EMap can quantify the grade of emotion based on common indices, such as "pleasantness" and "arousal", to the grade of change of facial expression patterns, and that emotion estimation can be performed to facial expressions for which two or more emotions are mixed.

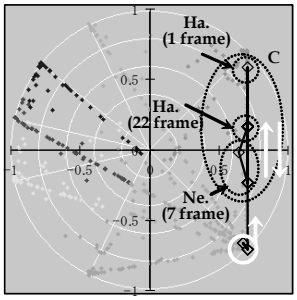




(b) State transition of Pattern 2.



(a) State transition of Pattern 1.



(c) State transition of Pattern 3.

Fig. 12. State transition on EMap (Pattern 1, 2 and 3 in Fig.11).

7. Conclusion

The study described in this chapter examined the method of generating a subject-specific emotion feature space for estimating the grade of emotion. The essential results obtained in this chapter are the following.
1) Hierarchical use of SOM with a narrow mapping space enables extraction of subject-specific expression categories.
2) The grade of emotions with "pleasantness" and "arousal" as indices can be matched to the grade of change of facial expression patterns on an EMap that is generated using the proposed method.
3) An EMap enables quantification of the grade of emotion to the grade of change of facial expression patterns, and to conduct emotion estimation to mixed facial expressions.
Objective evaluation on the relationship between the temporal change of facial expression patterns and state transition on an EMap accompanied by the context of a scene will be performed in a future study using natural facial expressions during conversation.

8. Acknowledgments

This work was supported in part by Ministry of Education, Culture, Sports, Science and Technology (MEXT) Grants-in-Aid for Young Scientists (B): No. 18700192 and No. 20700174.
This chapter is based on "Generation of Emotional Feature Space based on Topological Characteristics of Facial Expression Images," by M. Ishii, K. Sato, H. Madokoro and M. Nishida, which appeared in the Proceedings of the IEEE International Conference on Automatic Face and Gesture Recognition, (FG 2008), Amsterdam, The Netherlands, September 2008. © 2008 IEEE.

9. References

Akamatsu, S. (2002a). Recognition of Facial Expressions by Human and Computer [I]: Facial Expressions in Communications and Their Automatic Analysis by Computer, The Journal of the Institute of Electronics, Information, and Communication Engineers, Vol.85, No.9, pp.680-685. (in Japanese)

Akamatsu, S. (2002b). Recognition of Facial Expressions by Human and Computer [II]: The State of the Art in Facial Expression Analysis-1; Automatic Classification of Facial Expressions, The Journal of the Institute of Electronics, Information, and Communication Engineers, Vol.85, No.10, pp.766-771. (in Japanese)

Akamatsu, S. (2002c). Recognition of Facial Expressions by Human and Computer [III]: The State of the Art in Facial Expression Analysis-2; Recognition of Facial Actions, The Journal of the Institute of Electronics, Information, and Communication Engineers, Vol.85, No.12, pp.936-941. (in Japanese)

Akamatsu, S. (2003). Recognition of Facial Expressions by Human and Computer [IV: Finish]: Toward Computer Recognition of Facial Expressions Consistent with the Perception by Human, The Journal of the Institute of Electronics, Information, and Communication Engineers, Vol.86, No.1, pp.54-61. (in Japanese)

Fasel, B., & Luettin, J. (2003). Automatic Facial Expression Analysis: A Survey, Pattern Recognition, Vol.36, pp.259-275.

Gross, R. (2005). Face Databases, Handbook of Face Recognition, S.Li and A.Jain, ed., Springer-Verlag.

Kohonen, T. (1995). Self-Organizing Maps, Springer Series in Information Sciences.

Lienhart, R. & Maydt, J. (2002). An Extended Set of Haar-like Features for Rapid Object Detection, Proc. IEEE Int. Conf. Image Processing, Vol.1, pp.900–903.

Nielsen, R.H. (1987). Counterpropagation Networks, Applied Optics, vol.26, No.23, pp.4979-4984.

Pantic, M. & Rothkrantz, L.J.M. (2000). Automatic Analysis of Facial Expressions: The State of the Art, IEEE Trans. Pattern Analysis and Machine Intelligence, Vol.22, No.12, pp.1424-1445.

Pantic, M.; Valstar, M.F., Rademaker, R. & Maat, L. (2005). Webbased Database for Facial Expression Analysis, Proc. IEEE Int. Conf. Multimedia and Expo, pp.317-321.

Russell, J.A. & Bullock, M. (1985). Multidimensional Scaling of Emotional Facial Expressions: Similarity from Preschoolers to Adults, J. Personality and Social Psychology, Vol.48, pp.1290–1298.

Tian, Y.L.; Kanade, T. & Cohn, J.F. (2001). Recognizing Action Units for Facial Expression Analysis, IEEE Trans. Pattern Analysis and Machine Intelligence, Vol.23, No.2, pp.97-116.

Yamada, H. (2000). Models of Perceptual Judgments of Emotion from Facial Expressions, Japanese Psychological Review, Vol.43, No.2, pp.245–255.

19

Fingerprint Matching with Self Organizing Maps

Anastasia N. Ouzounoglou[1], Pantelis A. Asvestas[2] and George K. Matsopoulos[1]
[1]*School of Electrical and Computer Engineering*
National Technical University of Athens
Greece
[2]*Department of Medical Instruments Technology*
School of Technological Applications
Technological Educational Institute of Athens
Greece

1. Introduction

Fingerprint, namely the reproduction of a fingertip epidermis, produced when a finger is pressed against a smooth surface, is a human characteristic that has been systematically used for identification purposes for over 100 years.

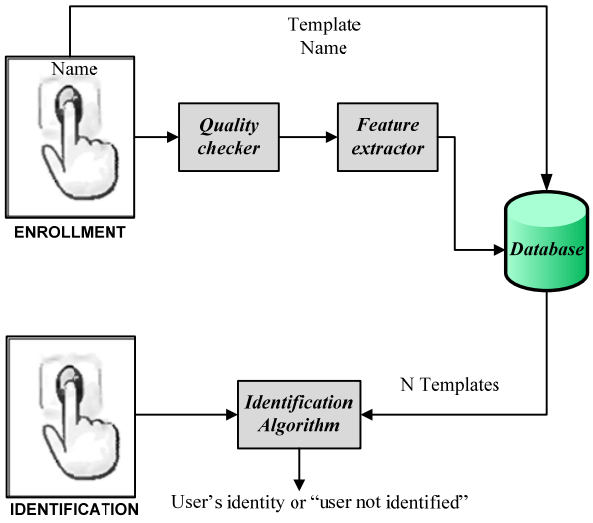


Fig. 1. Typical diagram of the fingerprint enrollment and identification processes.

The identification involves the comparison, also known as fingerprint matching, of an input fingerprint image with a template image stored in a database and either the calculation of a matching score or the extraction of a binary decision (mated/non-mated) (Fig. 1).
Usually, a matching algorithm does not operate directly on grayscale fingerprint images but requires the derivation of an intermediate fingerprint representation by means of a feature extraction stage. The features that are used for fingerprint representation can be broadly categorized as follows (Yager & Amin, 2004):

- features that are global characteristics of a fingerprint such as ridge flow
- features that refer to minutiae, such as ridge bifurcations and endings
- features that include all dimensional attributes of the ridge such as ridge path deviation, width, shape, pores, edge contours, breaks and scars.

The majority of the fingerprint identification systems are minutiae-based, but recently non-minutiae based systems as well as systems that use a combination of the features have been developed.
Jea et al, propose a fingerprint recognition system based only on minutiae matching (Jea & Govindaraju, 2005). This method is satisfactory for partial fingerprint images, where the core points or other features cannot be fully estimated. However it produces poor results compared to other methods that make use of more features of the fingerprint. In (Chan et al, 2004), a fast verification method has been developed that is based on the matching of minutiae located in a region centered at a reference (core) point of the fingerprint. This method is fast enough since only minutiae in a subregion of the fingerprint image are used, although it is most of the times hard to calculate the reference point especially for low quality fingerprint images. Another minutiae-based method has been developed by (He et al, 2006). This method introduces minutiae and local ridge information in fingerprint representation and both features are used in the fingerprint verification process. (Gu et al, 2006) proposed a fingerprint matching technique that combines both the model – based orientation field and the minutiae. Jain et al made use of all three levels of fingerprint features in the fingerprint matching process (Jain et al, 2006). Furthermore, some trials have been conducted in order to cope with non-linear distortions in fingerprint matching. (Chen et al, 2006) addressed a fingerprint matching and verification method based on normalized fuzzy similarity measure. According to this method, the input and template fingerprint images are aligned used a minutiae-based method and then the similarity between the two images is assessed using a novel similarity measure based on fuzzy theory. (Ross et al,2006) used a thin-plate spline (TPS) function for the estimation of an "average" deformation model based on ridge curve correspondences for a specific finger, when several impressions of that finger are available.
Since the extraction of minutiae is a difficult and time consuming task (Jain et al, 1997), some trials have been made for the development of fingerprint registration systems without the use of minutiae (Jain et al, 2000; Liu et al, 2006). In (Jain et al, 2000), a reference point was detected and the fingerprint was tessellated around this point. Then, a feature vector was constructed using a bank of Gabor filters. This vector was called the "FingerCode". The matching between the two fingerprint images was conducted by estimating the Euclidean Distance between the FingerCodes. The drawback of this method is, as explained above, that the reference point is sometimes hard to be found and there are cases of destroyed or partial fingerprint images that cannot be detected at all. In the method proposed in (Liu et al, 2006), the orientation field of the fingerprint images was estimated initially and then the

registration of the two images was carried out by maximization of the mutual information of the orientation fields. This method produced satisfactory results when the orientation field had been estimated in great detail, which is not always possible for low-quality fingerprint images.

This chapter proposes a minutiae-based fingerprint registration method. Initially, minutiae of the template image are extracted and validated. Then, the corresponding minutiae in the input image are determined by means of the proposed SOM-based algorithm. The obtained pairs of corresponding minutiae are used for the calculation of a matching score. The main advantages of the proposed method are that minutiae are extracted only on the template image and it is error-tolerant regarding the precise estimation of these points.

2. Methodology

In this chapter, an Automatic Fingerprint Identification Scheme (AFIS) is presented. The scheme comprises a series of processes as shown in the block diagram of Fig. 2. Without loss of generality, hereafter we denote the image of the fingerprint acquired during enrollment as the template (I_T) and the representation of the fingerprint to be matched as the input image (I_{inp}).

According to Fig. 2, the following processes are applied consequently only on the template image I_T:

- Ridge Enhancement
- Binarization
- Background Removal
- Thinning
- Minutiae Extraction and Validation
- Minutiae Automatic Correspondence.

In the following analysis, these processes are described in more details:

2.1 Ridge Enhancement

The ridges of the template image I_T (Fig. 3a) are enhanced using the method of oriented diffusion (Hastings, 2007), which is a recursive procedure. At each iteration *m*, the enhanced image, $I_{enh}^{(m)}$, is obtained by means of the following equation:

$$I_{enh}^{(m)}(x,y) = I_{enh}^{(m-1)}(x,y) + \gamma \frac{\partial^2 I_{enh}^{(m-1)}(x,y)}{\partial^2 \theta} \tag{1}$$

for $m = 1,2,\ldots,M$, where $I_{enh}^{(0)} = I_T$, $\theta = \theta(x,y)$ is the ridge orientation at pixel location (x,y) and γ is a constant. The ridge orientation of a fingerprint image, *I*, is estimated using the following procedure: initially the image is divided into blocks of size $W \times W$ and the gradients $\frac{\partial I(x,y)}{\partial x} = I_x(x,y)$ and $\frac{\partial I(x,y)}{\partial y} = I_y(x,y)$ are computed for each pixel (x,y). The

average gradient direction $\Phi(x,y)$ at each block centered at each pixel (x,y) is estimated as follows (Bazen et al, 2007), (Pratt, 1989):

$$
\begin{aligned}
V_x(x,y) &= \sum_{u=x-\frac{w}{2}}^{x+\frac{w}{2}} \sum_{v=y-\frac{w}{2}}^{y+\frac{w}{2}} 2 I_x(u,v) I_y(u,v) \\
V_y(x,y) &= \sum_{u=x-\frac{w}{2}}^{x+\frac{w}{2}} \sum_{v=y-\frac{w}{2}}^{y+\frac{w}{2}} \left[I_x^2(u,v) - I_y^2(u,v) \right] \\
\Phi(x,y) &= \frac{1}{2} \tan^{-1} \left(\frac{V_y(x,y)}{V_x(x,y)} \right)
\end{aligned}
\tag{2}
$$

with $-\frac{\pi}{2} < \Phi(x,y) \leq \frac{\pi}{2}$.

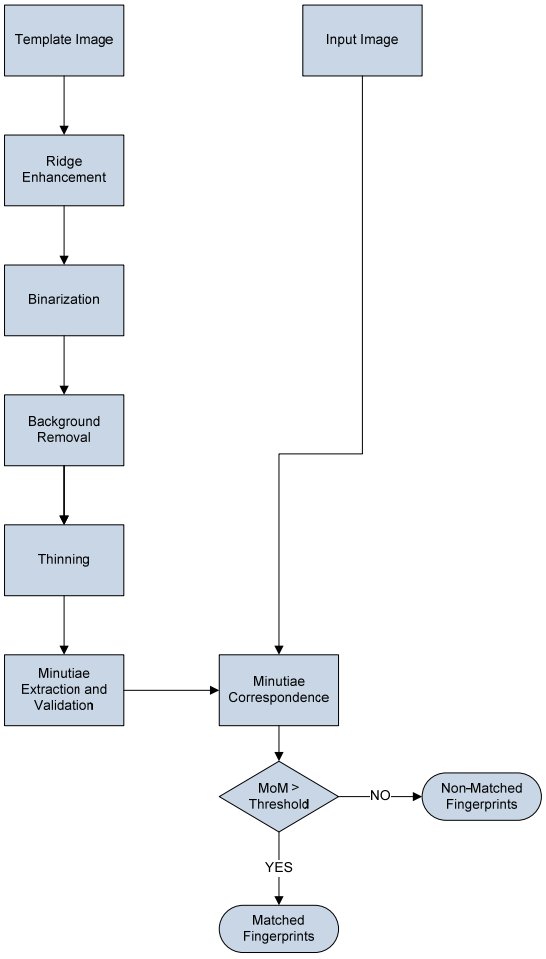


Fig. 2. Block diagram of the proposed automatic fingerprint identification scheme.

Then, the orientation field $O(x,y)$, is defined as follows:

$$O(x,y)=\begin{cases}\Phi(x,y)+\frac{1}{2}\pi & for \quad \Phi(x,y)\le 0\\ \Phi(x,y)-\frac{1}{2}\pi & for \quad \Phi(x,y)>0\end{cases} \tag{3}$$

with $-\frac{\pi}{2}<O(x,y)\le\frac{\pi}{2}$.

Due to the presence of noise, the local ridge orientation may not always be correct. Thus, the orientation field is smoothed using a unit integral filter with size $w_\Phi \times w_\Phi$. Firstly, the orientation filed is converted into a continuous vector field: $O_x(x,y)=\cos(2\cdot O(x,y)), O_y(x,y)=\sin(2\cdot O(x,y))$, where $O_x(x,y),\ O_y(x,y)$ are the x and y components of the vector field, respectively. Then, the continuous vector field is filtered by applying a low-pass filter as follows:

$$\begin{aligned}O_x^{'}(x,y)&=\sum_{u=-w_\Phi/2}^{w_\Phi/2}\sum_{v=-w_\Phi/2}^{w_\Phi/2}H(u,v)\cdot O_x(x-u\cdot w,y-v\cdot w)\\ O_y^{'}(x,y)&=\sum_{u=-w_\Phi/2}^{w_\Phi/2}\sum_{v=-w_\Phi/2}^{w_\Phi/2}H(u,v)\cdot O_y(x-u\cdot w,y-v\cdot w)\end{aligned} \tag{4}$$

where H is a two-dimensional low-pass filter with unit integral.

The smoothed orientation field of the fingerprint image at each pixel (x,y) is computed using:

$$\theta(x,y)=\frac{1}{2}\tan^{-1}\left(\frac{O_y^{'}(x,y)}{O_x^{'}(x,y)}\right) \tag{5}$$

The second derivative in (1) is calculated along an axis that forms an angle $\theta(x,y)$ with the x-axis using the following equation:

$$\begin{aligned}\frac{\partial^2 I_{enh}^{(m-1)}(x,y)}{\partial\theta^2}=&\frac{\partial^2 I_{enh}^{(m-1)}}{\partial x^2}\cdot\cos^2\theta(x,y)+\frac{\partial^2 I_{enh}^{(m-1)}}{\partial y^2}\cdot\sin^2\theta(x,y)+\\ &2\frac{\partial^2 I_{enh}^{(m-1)}}{\partial x\partial y}\cdot\cos\theta(x,y)\cdot\sin\theta(x,y)\end{aligned} \tag{6}$$

The output of this step is denoted as I_{enh} (Fig. 3b).

2.2 Binarization

For the binarization of the resulting enhanced template image, I_{enh}, the second step of the method described in (Hastings, 2007) is applied. The second derivative of the filtered image

is estimated in a direction normal to the orientation field. The second derivative is estimated using (6), but for $\theta' = \theta + \frac{\pi}{2}$. The sign of the second derivative is then examined based on the fact that the second derivative of a function is negative in the area of a local maximum and positive in the area of a local minimum. The output of this step is denoted as I_{BW} (Fig. 3c).

2.3 Background Removal

The first step for the estimation of the background of the binary image, I_{BW}, involves the partition of the image domain into blocks of *W*×*W* pixels. For each block, the mean value and the variance are calculated. If the variance is above a threshold, the pixels of this block are characterized as foreground pixels and the remaining ones as background pixel. Then only the foreground pixels of the binarized fingerprint image are maintained and the background pixels are set to the value 255. The output of this step is denoted as I_{BR} (Fig. 3d).

2.4 Thinning

The thinning process is applied on the negative version of I_{BR}. The two-subiteration thinning algorithm described in (Guo & Hall, 1989) is applied in order to get a thinned version of the ridge fingerprint image. This algorithm uses a 3×3 thinning operator in the area of pixel *P* as shown below:

$$\begin{matrix} P_1 & P_2 & P_3 \\ P_8 & P & P_4 \\ P_7 & P_6 & P_5 \end{matrix} \tag{7}$$

Let $C(P)$ $N(P)$, $N_1(P)$ and $N_2(P)$ be defined as follows:

$$C(P) = \bar{P}_2 \wedge (P_3 \vee P_4) + \bar{P}_4 \wedge (P_5 \vee P_6) + \bar{P}_6 \wedge (P_7 \vee P_8) + \bar{P}_8 \wedge (P_1 \vee P_2) \tag{8}$$

$$N(P) = \min[N_1(P), N_2(P)] \tag{9}$$

$$N_1(P) = (P_1 \vee P_2) + (P_3 \vee P_4) + (P_5 \vee P_6) + (P_7 \vee P_8) \tag{10}$$

$$N_2(P) = (P_2 \vee P_3) + (P_4 \vee P_5) + (P_6 \vee P_7) + (P_8 \vee P_1) \tag{11}$$

where the symbols, $\vee$ $\wedge$ and $\bar{}$ refer to logical OR, logical AND and logical complement, respectively.

A pixel that belongs to a ridge is removed from the ridge if the following three conditions are satisfied:

1. $C(P) = 1$
2. $2 \le N(P) \le 3$
3. In odd iterations: $(P_2 \vee P_3 \vee \tilde{P}_5) \vee P_4 = 0$
   In even iterations: $(P_6 \vee P_7 \vee \tilde{P}_1) \wedge P_8 = 0$

The algorithm stops when no new ridge pixels are deleted. The output of this step is denoted as I_{TH} (Fig. 3e).

2.5 Minutiae Extraction and Validation

The minutiae extraction method used in this work is based on the calculation of the Crossing Number (*CN*) (Maltoni et al, 2009). This method uses a 3×3 window in the local area of each ridge pixel *P*. The pixels in the area of *P* are numbered as follows:

$$\begin{matrix} P_4 & P_3 & P_2 \\ P_5 & P & P_9 \\ P_6 & P_7 & P_8 \end{matrix} \tag{12}$$

The *CN* for a ridge pixel *P* is:

$$CN = 0.5\sum_{i=1}^{8}\left|P_i - P_{i+1}\right|, \qquad P_9 = P_1 \tag{13}$$

If *CN* is one for a ridge pixel then the pixel is characterized as a ridge ending and if *CN* is three as a ridge bifurcation. However, the *CN* algorithm gives rise to many false minutiae. Therefore, a minutiae validation algorithm is applied (Tico & Kuosmanen, 2000). The algorithm uses an area of size *K*×*K* around each candidate minutiae. The central pixel of this area corresponds to the minutiae and is given the value -1. The rest points are initialized with the value zero. The steps of validation for a candidate bifurcation point are as follows:

1. The three pixels that are connected to the bifurcation point are labeled with values *l* = 1, 2 and 3 in a clockwise direction.
2. Three ridge branches, that originate from the three eight-connected to the minutiae pixels, are labeled with the corresponding value of *l*.
3. The number of transitions from 0 to 1, 0 to 2 and 0 to 3 is counted in a clockwise direction along the border of the selected area. If all three transitions are equal to one then the point is characterized as a bifurcation point.

In Fig. 3f, the extracted minutiae of the template image are shown.

It must be noted that in this work only the minutiae that correspond to bifurcation points are used in the process of registration, while endings are not used at all. That is based on the fact that the number of bifurcation points of the images of the test database is from 14 to 25; that is an adequate number of neurons for the SOM algorithm. The number of ending minutiae points is from 32 to 59. If the ending points were to be used, the execution time of the algorithm would be larger without producing better results. Furthermore, SOM algorithm works better if the points are scattered in the image, something that is valid for bifurcation points.

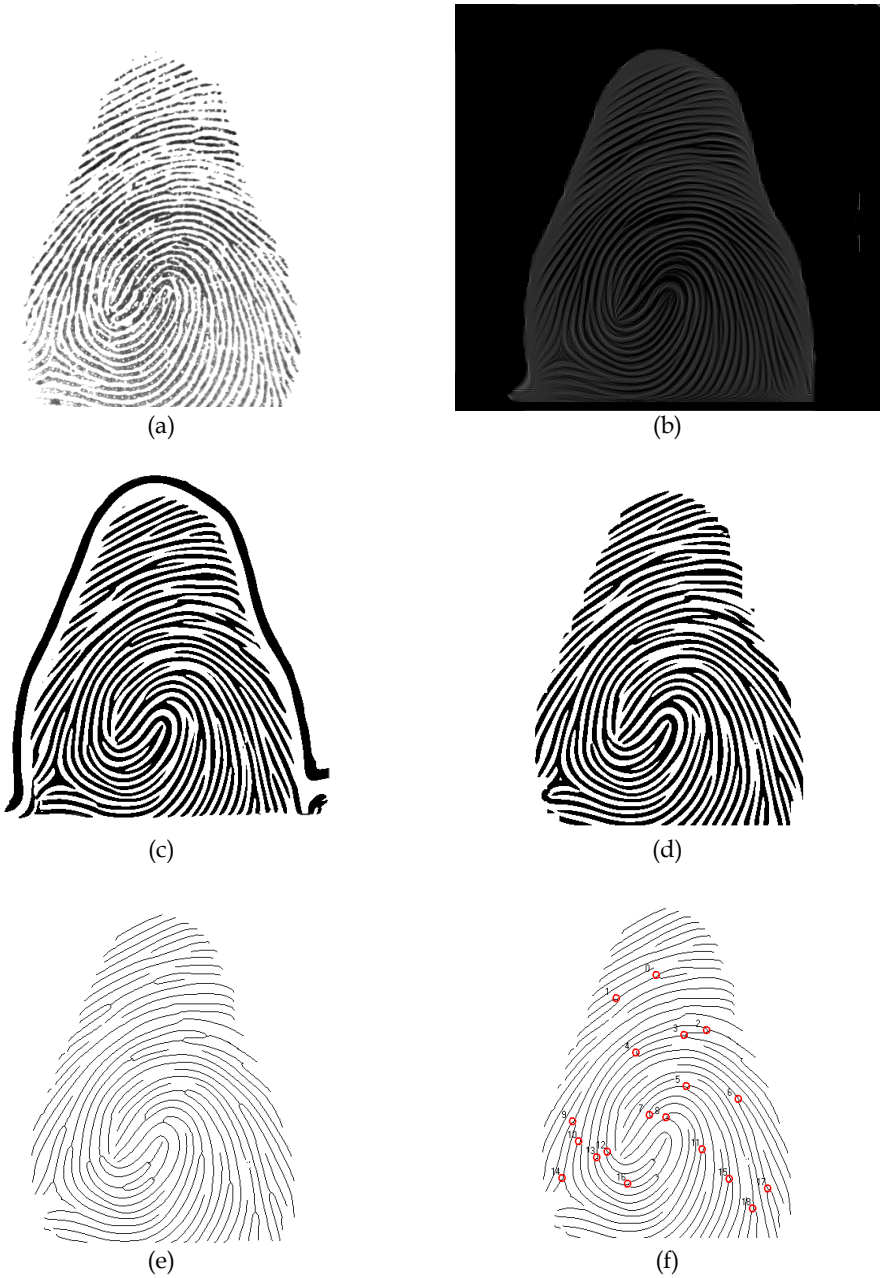


Fig. 3. Output images of the processes of the proposed automatic fingerprint identification scheme. (a) Template image. (b) Ridge enhanced image. (c) Binarized image. (d) Binarized image after background removal. (e) Thinned image. (f) Minutiae after the application of the extraction and validation process.

2.6 Minutiae Correspondence based on SOM algorithm

An automatic method for establishing minutiae correspondences between the template image and the input image is applied based on the theory of the SOMs. The SOMs is a neural network algorithm, which uses a competitive learning technique to train itself in an unsupervised manner. Kohonen firstly established the relevant theory and explored possible applications (Kohonen, 2000). The Kohonen model comprises a layer of *N* neurons, ordered usually in a one- or two-dimensional grid. The training of the network is performed in an iterative way. At each iteration n, an input vector $\mathbf{x} \in \Re^m$ is presented to the network; the neuron *j* with weight vector $\mathbf{w}_j \in \Re^m$ is declared as the winning neuron, according to the following rule:

$$j = \arg\min_i \left(\left\| \boldsymbol{x} - \boldsymbol{w}_i \right\| \right) \tag{14}$$

The weight vectors of the winning neuron *j* and its neighboring neurons *i* are updated as follows:

$$\mathbf{w}_i(n+1) = \mathbf{w}_i(n) + h_{ij}(n)\left[\mathbf{x}(n) - \mathbf{w}_i(n) \right] \tag{15}$$

where $h_{ij}(n) = h\left(\left\| \mathbf{r}_i - \mathbf{r}_j \right\|, n \right)$ is a kernel defined on the neural network space as a function of the distance $\left\| \mathbf{r}_i - \mathbf{r}_j \right\|$ between neurons *i* and *j* and the iteration number *n*. This kernel is a Gaussian function, which has maximum value at inter-neuron distance in the case of $i = j$. The width of this function decreases monotonically with the iteration number. In this way, convergence to the global optimum is attempted during the early phases of the self-training process, whereas gradually the convergence becomes more local as the size of the kernel decreases.

Before proceeding to the analytical description of the network topology, some notations must be introduced. Let $\mu_A(I) = I(x,y), (x,y) \in A \subset Z^2$ denote the restriction of an image *I* in the region A and $\mathbf{T}_{\mathbf{w}}(A) : Z^2 \to \Re^2$ be a transformation with parameters $\mathbf{w} = (w_1, w_2, ..., w_d)$ of the region A, where d is the number of parameters needed for the definition of the specific transformation. In this chapter, the similarity transformation was considered defined as follows:

$$x' = r\cos\theta \cdot x - r\sin\theta \cdot y + dx$$
$$y' = r\sin\theta \cdot x + r\cos\theta \cdot y + dy$$

with parameter vector $\mathbf{w} = (r, dx, dy, \theta)$ and dx, dy, r and θ is the horizontal displacement, the vertical displacement, the scaling and the angle of rotation, respectively. The value of each parameter of the used transformation is bounded according to the following inequalities:

$$|dx| \le dx_{\max}$$
$$|dy| \le dy_{\max}$$
$$|\theta| \le \theta_{\max}$$
$$|r| \le r_{\max}$$

where $dx_{\max}, dy_{\max}, r_{\max} > 0$ and $0 < \theta_{\max} < \pi$. Furthermore, $MoM(I_1, I_2)$ denotes a measure of match between two images I_1 and I_2. Without loss of generality, it is assumed that $MoM(I_1, I_2)$ lies in the range $[0,1]$ where 1 indicates perfect matching and 0 indicates no matching.

Let $\mathbf{P}_i = (x_i, y_i)$, $(i = 1,2,...,N)$ be the minutiae extracted from the template image, then the algorithm places a neuron on each one of them. Each neuron is associated with a square area $A_i = [x_i - R, x_i + R] \times [y_i - R, y_i + R]$ of $(2R+1)^2$ pixels centered at the position of the neuron. Additionally, a weight vector $\mathbf{w}_i$, which holds the parameters of a local transformation (rigid or similarity), is assigned to each neuron.

The SOM network is trained as follows:

1 For each neuron, the components of the weight vector are initialized to default values (i.e. $w_i(0) = (0,0,0)$ for rigid transformation and $w_i(0) = (1,0,0,0)$ for similarity transformation) and the quantities $MoM_i(0) \equiv MoM\left(\mu_{A_i}(I_T), \mu_{\mathbf{T}_{\mathbf{w}_i(0)}(A_i)}(I_{inp})\right)$ are calculated, the variable MoM_{best} is set to a very large (in magnitude) negative value and the iteration variable, *n*, is set to 1.

2 While *n* is less than $n_{\max}$:

- If the average value of $MoM_i(n-1)$, $MoM_{ave}(n-1) = \frac{1}{N}\sum_{i=1}^{N} MoM_i(n-1)$, is better than MoM_{best}, then $MoM_{best} = MoM_{ave}(n-1)$ and the current weights are stored as $\mathbf{w}_i (i = 1,2,\ldots,N)$.
- An *input vector*, $\mathbf{s}(n)$, is generated pseudo-randomly.
- For every neuron, the quantity $MoM_i(n) = MoM\left(\mu_{A_i}(I_T), \mu_{\mathbf{T}_{\mathbf{s}(n)}(A_i)}(I_{inp})\right)$ is calculated.
- The *winning neuron*, k_n, in the current iteration, is defined as $k_n = \arg\max_i \left(MoM_i(n)\right)$ under the condition $MoM_{k_n}(n) > MoM_{ave}(n-1)$.
- The weights of the neurons are updated according to the following equation:

$$\mathbf{w}_i(n) = \mathbf{w}_i(n-1) + h(k_n, n, i)[\mathbf{s}(n) - \mathbf{w}_i(n-1)] \tag{16}$$

where $h(k_n, n, i)$ $(i = 1,2,\ldots..N)$ is given by the following equation:

$$h(k_n, n, i) = \begin{cases} L^n, & \left\|\mathbf{P}_{k_n} - \mathbf{P}_i\right\| < a^n \cdot d_0 \\ 0, & \text{otherwise} \end{cases} \tag{17}$$

where $L, a, d_0 \in \Re$ are parameters to be defined later and $\| \ \|$ denotes the Euclidean norm.

- The iteration variable is increased by one.

The best value of the measure of match, MoM_{best}, provides an index of the matching between the two images.
Several issues regarding the proposed method should be discussed. First of all, in order to cope with the differences in contrast and/or brightness between the template and the input image, the selected measure of match was the squared correlation coefficient, namely:

$$MoM(I_1, I_2) = \frac{\left(\sum_{x,y} \left[I_1(x,y) - \overline{I}_1 \right] \left[I_2(x,y) - \overline{I}_2 \right] \right)^2}{\sum_{x,y} \left[I_1(x,y) - \overline{I}_1 \right]^2 \sum_{x,y} \left[I_2(x,y) - \overline{I}_2 \right]^2} \tag{18}$$

where $\overline{I}_1$ and $\overline{I}_2$ are the mean pixel value for image I_1 and I_2, respectively.
As was mentioned before, a input vector $\mathbf{s}(n)$ is generated pseudorandomly, according to the following relation:

$$\mathbf{s}(n) = \mathbf{w}_{k_n} + \boldsymbol{\upsilon} \tag{19}$$

where $\boldsymbol{\upsilon} = (\upsilon_1, \upsilon_2, \ldots, \upsilon_d)$ is a d-dimensional normally distributed random variable with mean vector $\underbrace{(0,0,\ldots,0)}_{d}$ and covariance matrix $diag(\sigma_1^2(n), \sigma_2^2(n), \ldots, \sigma_d^2(n))$. The standard deviation $\sigma_j(n)$ of the random variable υ_j ($j = 1,2,\ldots,d$) varies with the iteration variable as follows:

$$\sigma_j(n) = (U_j - L_j) e^{-pn} \tag{20}$$

where U_j (L_j) denotes the maximum (minimum) allowed value for the j-th component of the input vector and p determines the rate of exponential change of $\sigma_j(n)$. It must be noted that the above equations provide random signals which in general lie in the range $[w_{k_n,j} - (U_j - L_j), w_{k_n,j} + (U_j - L_j)]$. When a generated input vector is not in the allowed range $[L_j, U_j]$, then it is discarded and a new input vector is produced until $s_j(n) \in [L_j, U_j]$. The parameter $\sigma_j(n)$ controls how far from the weights of the current winning neuron the input vector can reach. As the iteration variable evolves, the magnitude of $\sigma_j(n)$ falls exponentially and the generated input signals are more localized around the weights of the current winning neuron. This is a desired property, since as the number of iterations grows, the weights of the current winning neuron get closer to the parameters of the solution of the registration problem.
The parameter d_0 provides the initial radius of a circular region around the winning neuron. Only neurons inside this region are updated. Usually, d_0 is set to the maximum distance between minutiae. As can be seen from (17), this distance is reduced with geometric rate determined by the parameter a ($0 < a \leq 1$). The parameter L acts like a gain constant for the magnitude of the update that is applied to the weights of the neurons. This parameter also

decreases geometrically as the iteration variable evolves. The range of values L is between 0.99 and 1.0.

It should be pointed out that a sufficient number of minutiae should be extracted in order to achieve an accurate registration result. Furthermore, the minutiae must be distributed over the whole image (if possible). The degree of sparseness of the bifurcation points can be determined by checking if the standard deviation of the $x-y$ coordinates is above a predefined threshold. Experiments have shown that for fingerprint images twelve minutiae, with standard deviation of the $x-y$ coordinates that exceeds 100, are sufficient in order to obtain accurate registration results. Since the transformed region $\mathbf{T}_{\mathbf{s}(n)}(A_i)$ does not have integer coordinates, bilinear interpolation (Press et al, 1992) is used in order to calculate $MoM_i(n)$.

3. Results

In order to assess the performance of the proposed automatic fingerprint identification scheme, the VeriFinger_Sample_DB database of fingerprint images was used (Neurotechnology, 2007). The database contains fingerprint images from nine different persons, for six fingers of each person and for eight impressions of each finger; thus 432 images in total. Each fingerprint image has size 504×480 pixels. Two different data sets were used. The first data set (SET I) consists of fingerprint images subject to known transformations, while the second one (SET II) comprises of fingerprint image pairs from the database (unknown transformation).

3.1 SET I: Data subject to known transformations

The data of SET I consist of 11 fingerprint images from the VeriFinger_Sample_DB database. Each image is considered as a template image and is transformed into two different input images according to two affine transformations (*Affine-1*, *Affine-2*); thus, a total of 22 image pairs were finally obtained for further testing. The affine transformation function is described as follows:

$$\begin{aligned} X &= s\cos\theta\left(x-x_{cm}\right)-s\sin\theta\left(y-y_{cm}\right)+t_1+x_{cm} \\ Y &= s\sin\theta\left(x-x_{cm}\right)+s\cos\theta\left(y-y_{cm}\right)+t_2+y_{cm} \end{aligned} \tag{21}$$

where $x(X)$, $y(Y)$ are the column and row indices, respectively, in the template (input) image and (x_{cm}, y_{cm}) are the coordinates of the geometric centre of the template image, s is the scaling parameter, θ the rotation angle and t_1, t_2 the translation along *x*-axis and *y*-axis, respectively. For *Affine-1,* the parameters used were: $s=1$, $\theta=5°$, $t_1=5$ and $t_2=5$, while for *Affine-2:* $s=1.05$, $\theta=10°$, $t_1=10$ and $t_2=10$.

After the sequential application of the processes described in Fig. 2, the number of the extracted minutiae in the template images of the data SET I varied from 11 to 30 points. The actual corresponding points, $\mathbf{Q}_i$, in the input image of each pair were obtained by transforming the minutiae of the template image according to (21).

The performance of the proposed automatic point correspondence detection algorithm was quantitatively assessed using the Root Mean Square Error (RMSE). The RMSE was calculated between the detected points $\tilde{\mathbf{Q}}_i$ and the actual corresponding points $\mathbf{Q}_i$, $i = 1, 2, ..., N$ according to the equation:

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left\|\mathbf{Q}_i - \tilde{\mathbf{Q}}_i\right\|^2} \tag{22}$$

The implementation of the algorithm described in the previous section assumes that a number of parameters are beforehand determined. The values of all parameters used are listed in Table 1. The same values of the parameters were applied to all images. It must be also noted that the proposed SOM algorithm was executed ten times for each image pair and the average value of the RMSE (in pixels) was calculated.

Equation	Description	Symbol	Value
(1)	Iterations of Oriented Diffusion (Ridge Enhancement)	M	100
(1)	Spread of Gaussian Filter (Ridge Enhancement)	γ	2
(2)	Block Size for Orientation Field	$w \times w$	16×16
(4)	Size of Unit Integral Filter (Orientation Field)	$w_\Phi \times w_\Phi$	3×3
	Block Size for Image Segmentation	$W \times W$	15×15
	Area Size for Minutiae Validation	$K \times K$	23×23
SOM Algorithm Parameters			
(17)	Initial Neighbourhood Size of Winning Neuron	d_0	max distance between neurons
(17)	Rate of Change of d_0	a	0.90
(17)	Gain constant for the magnitude of the update that is applied to the weights of the neurons - Learning Rate	L	0.995
(20)	Rate of change of input vector range	p	0.01
	Half Size of Square Region of each neuron	R	10
	Number of Iterations	$n_{\max}$	6,000
	Maximum Value of Scaling	$r_{\max}$	1.4
	Maximum Value of Horizontal Displacement (in pixels)	$dx_{\max}$	100
	Maximum Value of Vertical Displacement (in pixels)	$dy_{\max}$	100
	Maximum Value of Angle of Rotation (degrees)	$\theta_{\max}$	20

Table 1. Implementation parameters of the proposed automatic fingerprint identification scheme.

An example of the performance of the proposed algorithm in defining automatic correspondence is shown in Fig. 4 for typical fingerprint images of SET I. A template image

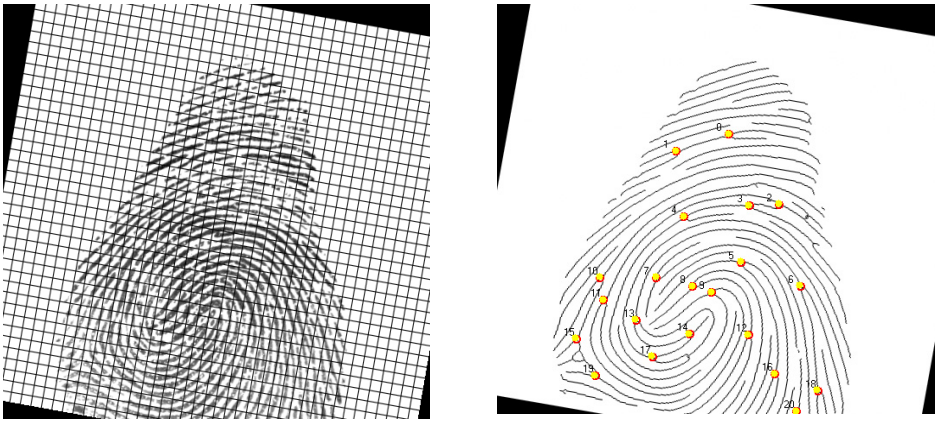
of the SET I is displayed in Fig. 4a along with its input images as transformed by the *Affine-1* (Fig. 4c) and *Affine -2* (Fig. 4e) transformations. The minutiae of the template thinned image are shown in Fig. 4b as red dots. The actual corresponding points, calculated by the *Affine-1* (Fig 4d) and *Affine-2* (Fig 4f) transformations of the minutiae in the template image according to (18), are marked with red dots, while the corresponding points detected by the SOM algorithm are marked with yellow dots in the same figures. From this figure, it is evident that the detected points by the proposed algorithm correctly match the actual corresponding points.

(a)

(b)

(c)

(d)



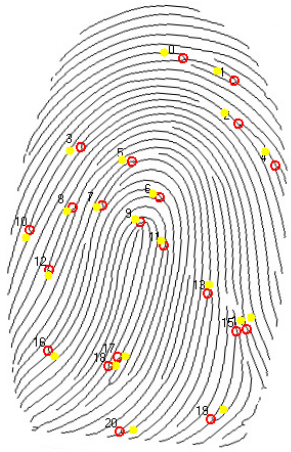
(e) (f)

Fig. 4. The performance of the proposed SOM algorithm in defining automatic point correspondence of a typical fingerprint image of SET I and its transformed images. Red dots indicate the actual corresponding points, while yellow dots indicate the detected points using the SOM algorithm. (a) Typical fingerprint template image. (b) Minutiae of the template image, after the application of the extraction and validation process. (c) Transformed template image using the *Affine-1* transformation. (d) Minutiae correspondences of the template and *Affine-1* transformed image using the SOM algorithm. (e) Transformed template image using the *Affine-2* transformation. (f) Minutiae correspondences of the template and *Affine-2* transformed image using the SOM algorithm.

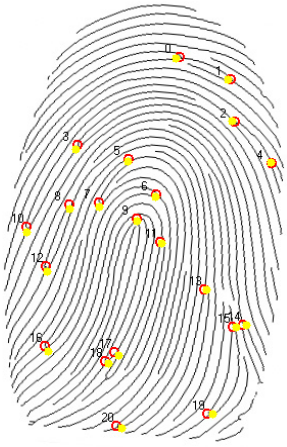
Furthermore, the performance of the proposed SOM algorithm is tested for a typical fingerprint image of SET I and its transformed image (*Affine-1* transformation) against the number of iterations. Fig. 5(a) depicts the initial position of the minutiae in the input image (before the training process), while Fig. 5(b)-(d) show three different phases of the network training, after 800, 2,800 and 6,000 iterations, respectively. A plot of the average RMSE, as well as of the best values of the MoM, between the detected points and the actual corresponding points with respect to the number of iteration is shown in the diagram of Fig. 5(e). As the number of iteration increases, the value of the average RMSE drops from its initial value of 11.267 pixels to the value of 5.197 pixels, after 800 iterations (Fig.5(b)) and to the value of 0.581 pixels after 2,800 iterations (Fig.5(c)). The value of the MoM increases from its initial value to the value of 0.11 after 800 iterations and to the value of 0.32 after 2,800 iterations. The average RMSE gets its best value (0.389 pixels) after 3,000 iterations (Fig.5(d)) and then it remains stable (up to 10,000 iterations), while the average MoM gets its best value of 0.94 after 6,000 iterations.

The performance of the SOM algorithm in terms of the average RMSE was also studied for different values of two important parameters of the algorithm: the parameter L and the parameter a for a typical image pair of SET I, as transformed by the *Affine-1*. The parameter L acts like a gain constant for the magnitude of the update that is applied to the weights of the neurons and the range of its value was between 0.99 and 1.0. The parameter a relates with the reduction rate of the distance of the initial radius around the winning neuron and the range of its values was between 0.1 and 1.0. During these experiments, all the other parameters of the SOM algorithm were set to values included in Table 1. As can be noticed

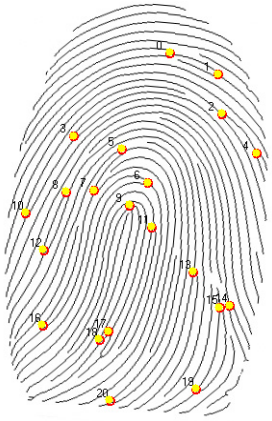
from Fig. 6a, the performance of the SOM algorithm is relative stable for different values of L, with lowest value of the average RMSE for $L = 0.995$. Similarly, from Fig. 6b, the best performance of the SOM algorithm in terms of average RMSE was obtained for $a = 0.90$.



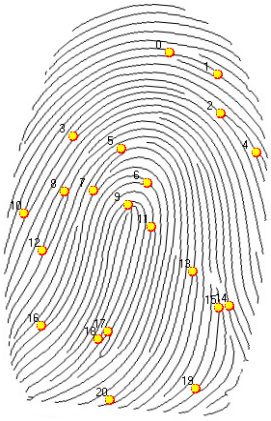
(a)



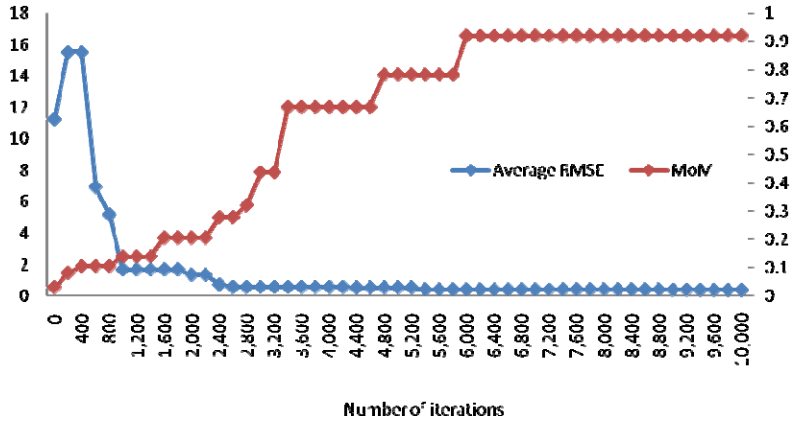
(b)



(c)



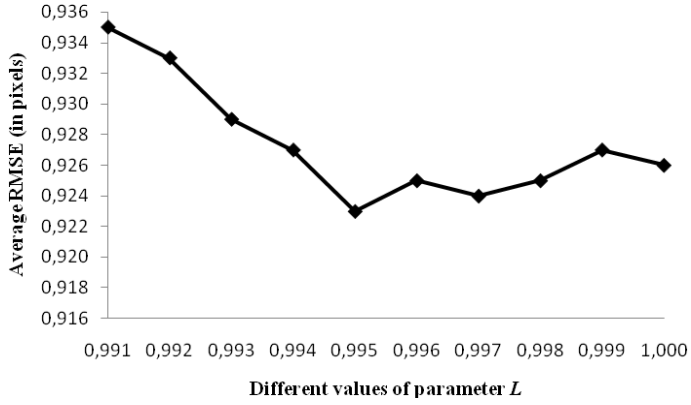
(d)



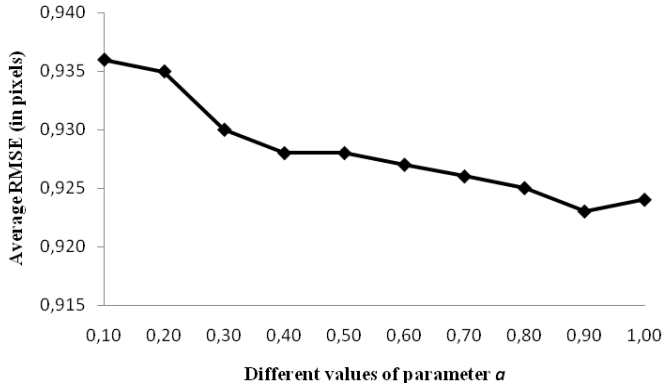
(e)

Fig. 5. The performance of the proposed SOM algorithm in defining automatic point correspondence for a typical fingerprint image of SET I and its transformed image (*Affine-1* transformation) against the number of iterations. Red dots indicate the actual corresponding points, while yellow dots indicate the detected points using the SOM algorithm. (a) Initial points position. (b)-(d) Correspondences obtained after 800, 2,800 and 6,000 iterations, respectively. (e) Performance of the SOM algorithm in terms of the average RMSE and the best value of MoM for the specific pair with respect to the number of iterations.

Quantitative results from the application of the proposed automatic fingerprint identification scheme to SET I are presented in Table 2 in terms of the best values of MoM and the average RMSE (in pixels). For the *Affine-1* transformation, the average RMSE lies between 0.389 and 0.478 pixels. For the *Affine-2* transformation, the average RMSE lies between 0.643 and 2.089 pixels.



(a)



(b)

Fig. 6. The performance of the proposed SOM algorithm in defining automatic point correspondence for a typical fingerprint image of SET I and its transformed image (*Affine-1* transformation) against (a) different values of the parameter *L* and (b) different values of the parameter *a*.

From Table 2, it is evident that the proposed algorithm is capable of determining correct correspondences with subpixel accuracy for the *Affine-1* transformation. However, as is expected, its performance depends on the kind of transformation applied to the image pairs. Thus, the value of the average RMSE deteriorates for *Affine-2* transformation. Yet again, the best values of the RMSE for the transformations remain relatively low, while the low values of the standard deviation indicate high reproducibility of the proposed algorithm for all transformations used.

Pairs of SET I	**Affine-1 transformation**		**Affine-2 transformation**	
	Average RMSE	**MoM**	**Average RMSE**	**MoM**
1	0.389	0.943	0.643	0.831
2	0.462	0.949	0.6999	0.863
4	0.448	0.941	1.317	0.689
5	0.478	0.945	0.746	0.679
6	0.471	0.919	1.805	0.562
7	0.448	0.897	2.089	0.540
8	0.437	0.912	1.414	0.463
9	0.390	0.954	0.805	0.729
10	0.439	0.950	0.808	0.791
11	0.404	0.912	0.722	0.805
Mean	**0.436**	**0.931**	**1.077**	**0.701**
STD	**0.0307**	**0.019**	**0.501**	**0.130**

Table 2. Quantitative results obtained by the proposed automatic fingerprint identification scheme to SET I in terms of best value of MOM and the average RMSE (in pixels).

3.2 SET II: Data subject to unknown transformations

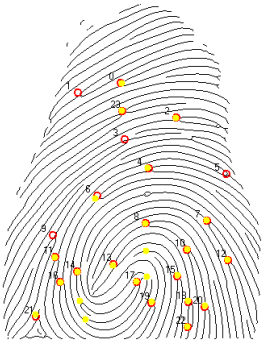
The SET II comprises of all the fingerprint images of the VeriFinger_Sample DB (Neurotechnology, 2007). This database consists of 408 fingerprint images. Tests have been carried out on 1,428 image pairs of same fingers and on 1,428 image pairs of different fingers. Each pair consists of a template and an input image, and the transformation that associates these two images is, unlike the previous case, unknown.

The proposed automatic fingerprint identification scheme was applied to all image pairs using the same values of the various parameters listed in Table 1. Since the actual corresponding points $\mathbf{Q}_i$, $i = 1, 2, ..., N$ cannot be directly calculated by transforming the minutiae of the template image, a different procedure for the visual evaluation of the proposed algorithm had to be applied. Initially, minutiae were extracted in both the template and input images, according to the aforementioned procedures. Then, new minutiae of the input image were obtained by the application of the SOM algorithm. These two sets of minutiae on the input image were visually compared.

In Fig. 7, the performance of the proposed automatic fingerprint identification scheme, including the SOM algorithm, is visually assessed for a fingerprint image pair of SET II. In Fig. 7a and 7b, the initial template image of a finger and the input image of the same figure, as obtained with different settings, are displayed. Fig. 7c displays the thinned input image along with its extracted minutiae (red dots) and the minutiae (yellow dots) as obtained by the application of the proposed scheme. It can be noticed that a satisfactory correspondence has been achieved for the majority of the minutiae. For this particular pair, the best value of MoM was 0.798.

(a)

(b)



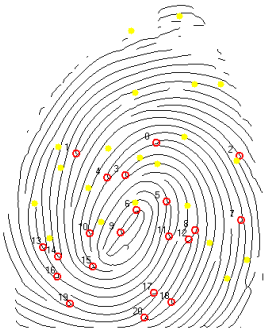
(c)

Fig. 7. The performance of the proposed SOM algorithm in defining automatic point correspondence for a fingerprint image pair of SET II (same finger under different acquisition settings). (a) Initial template image. (b) The input image. (c) Thinned input image along with its extracted minutiae (red dots) and the estimated minutiae as obtained by the proposed scheme (yellow dots).

In the case of applying the proposed automatic fingerprint identification scheme in pairs of different fingers, the results are considerably deteriorated, as it was expected. In Fig. 8a and 8b, two fingerprint images of different fingers are displayed. Fig. 8c shows the thinned image along with its extracted minutiae (red dots) and the minutiae (yellow dots), as obtained by the application of the proposed scheme, where it is evident the failure of minutiae correspondence. For this particular example, the best value of MOM was 0.145.

(a)

(b)



(c)

Fig. 8. The performance of the proposed SOM algorithm in defining automatic point correspondence for a fingerprint image pair of SET II (same finger of different person under different acquisition settings). (a) Initial template image. (b) The input image. (c) Thinned input image along with its extracted minutiae (red dots) and the estimated minutiae as obtained by the proposed scheme (yellow dots).

Different types of error rates are used as metrics for the operative capability of biometric authentication systems in general and for fingerprint image identification systems in particular. The result of a comparison in the feature matcher within a fingerprint image recognition system is called Matching Score. It measures the similarity between the fingerprint image and the stored template. If the value of Matching Score approaches 1 (if normalized between in the range [0,1]), the more likely that both fingerprints originate from the same finger. On the other hand, if Matching Score is near 0, it will be quite probable that both fingerprints are from different fingers.

The decision of the system is determined by threshold T, i.e. if Matching Score passed the threshold, the fingerprints are regarded as being of the same finger (matching pair) whereas if Matching Score is below the threshold, the fingerprints are regarded as being different (non-matching pair). The performance of a biometric system is assessed by means of the False Acceptance Rate (FAR), False Rejection Rate (FRR) and Equal Error Rate (EER) for various values of the threshold:

$$\begin{aligned} FAR(T) &= \frac{\text{Number of comparisons of different fingers resulting in a match}}{\text{Total number of different fingers}} \\ FRR(T) &= \frac{\text{Number of comparisons of the same fingers resulting in a non-match}}{\text{Total number of the same fingers}} \\ ERR &= FAR(T^*) \end{aligned} \tag{23}$$

where T^* is the value of the threshold, such that $FAR(T^*) = FRR(T^*)$.

The FAR and FRR were calculated by thresholding the best MoM obtained by the SOM-based algorithm for the data of SET II. The corresponding curves are shown in Fig. 9. The

intersection point of the two curves corresponds to the *ERR*. The value of the ERR is 0.08 and obtained for $T^* = 0.219$.

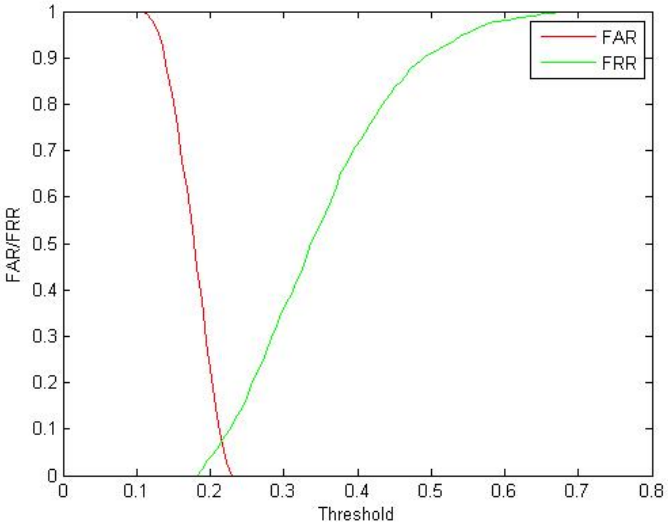


Fig.9 The FAR/FRR curves for various threshold values of the MoM. The intersection point corresponds to the ERR.

4. Conclusion

In this chapter, a SOM-based algorithm for fingerprint identification is presented. The minutiae of the template image are used as neurons of a neural network and the proposed algorithm detects the set of minutiae in the input image in an iterative way. The main advantage of this method, against other minutiae-based fingerprint recognition methods, is that the minutiae of only the template image have to be estimated. Furthermore the method is error-tolerant in the estimation of the minutiae of the template image. This is a matter of high importance since the precise estimation of minutiae is a difficult task, especially for low-quality fingerprint images. The overall performance of the proposed method was 92%.
It should be noted that the focus of the chapter was to show the feasibility of the SOM theory for registering fingerprint images. The proposed implementation of the SOM model could be considered as a method for finding the optimum value of an objective function. Under this framework, the proposed registration scheme provides several "degrees of freedom" regarding its parameters. For example, another measure of match (such as mutual information) could be used, other characteristic points of the fingerprint images (such as the points with high value of deviation) could be used, minutiae could be extracted by means of other methods. Moreover the algorithm has not been tested on partial fingerprint images or images with high degree of distortion.

5. References

Bazen A. & Gerez, S. (2002).Systematic Methods for the Computation of the Directional Fields and Singular Points of Fingerprints, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 24, Issue 7, Jul. 2002,pp. 905-919,ISSN:0162-8828.

Chan, K. C. ; Moon, Y. S. & Cheng P.S. (2004). Fast Fingerprint Verification Using Subregions of Fingerprint Images, *IEEE Transactions on Circuits and Systems For Video Technology.*, Vol.14, Issue 1, Jan. 2004,pp. 95-101,ISSN:1051-8215.

Chen, X.; Tian, J. & Yang, X. (2006). A New Algorithm for Distorted Fingerprints Matching Based on Normalized Fuzzy Similarity Measure, *IEEE Transactions on Image Processing*, Vol. 15, Issue 3, March 2006,pp. 767-776, ISSN: 1057-7149

Gu, J.; Zhou, J. & Yang, Ch.(2006). Fingerprint Recognition by Combining Global Structure and Local Cues, *IEEE Transactions on Image Processing*, Vol.15, Issue 7, Jul. 2006,pp.1952-1964, ISSN: 1057-7149.

Guo,Z.; & Hall,R; Parallel Thinning with Two-Subiteration Algorithms,Communications , Association of Computer Machinery New York USA,Vol. 32 Issue 3,pp. 359-373,1989,ISSN:0001-0782.

Hastings,R.(2007). Ridge Enhancement in Fingerprint Images Using Oriented Diffusion, Proceedings of the 9th Biennial Conference of the Australian Pattern Recognition Society on Digital Image Computing Techniques and Applications, pp.245-252, ISBN: 0-7695-3067-2, Dec. 2007,IEEE Computer Society, Washington DC-USA

He, Y. ; Tian, J. ; Li, L. ; Chen, H. & Yang, X. (2006). Fingerprint Matching Based on Global Comprehensive Similarity, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 28, Issue 6, Jun. 2006,pp. 850-862 ISSN:0162-8828.

Hong, L.; Wan,Y. & Jain,A. Fingerprint Image Enhancement: Algorithm and Performance Evaluation, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 20, Issue 8, Aug. 1998,pp. 777-789,ISSN: 0162-8828.

Jain A., Hong, L. & Bolle, R.(1997). On – Line Fingerprint Verification, *IEEE Trans Pattern Analysis and Machine Intelligence*, Vol. 19, Issue 4, Apr. 1997,pp. 302-314,ISSN: 0162 - 8828.

Jain, A.; Prabhakar, S.; Hong, L. & Pankanti, Sh.(2000). Filterbank-Based Fingerprint Matching, *IEEE Transactions on Image Processing,* Vol. 9,Issue 5, May 2000,pp. 846-859.

Jain, A.K; Chen, Y. & Demirkus, M. (2007). Pores and Ridges: High-Resolution Fingerprint Matching Using Level 3 Features, *IEEE Transactions on Pattert Analysis and Machine Intelligence*, Vol.29, Issue 1, Jan.2007,pp. 15-27.

Jea, T-Y. & Govindaraju, V. (2005). A minutia-based partial fingerprint recognition system. *Elsevier Pattern Recognition*, Vol. 38, Issue 10, Oct. 2005 pp. 1672– 1684

Kohonen,T.(2000). *Self-Organizing Maps, 3rd Edition*., Springer-Verlag, ISBN:3540679219, Berlin, Germany.

Liu, L. ; Jiang, T.; Yang,J. & Zhu, Ch.(2006) Fingerprint Registration by Maximization of Mutual Information, *IEEE Transactions on Image Processing*, Vol. 15, Issue 5, May 2006,pp.1100-1110,ISSN:1057-7149.

Maltoni,D.; Maio,D; Jain,A.K. & Prabhakar,S(2009) "Handbook of Fingerprint Recognition,Second Edition", Springer, ISBN:978-1-84882-253-5,London.

Matsopoulos,G.K; Asvestas,P.A; Mouravliansky,N.A. & Delibasis,K.K.(2004) Multimodal registration of retinal images using Self Organizing Maps, *IEEE Transactions on Medical Imaging,* Vol. 23, Issue 12, Dec. 2004,pp. 1557–1563, ISSN:0278-0062.

Pratt, W.(2001), *Digital Image Processing, Third Edition,* Wiley-Interscience Publication, ISBN: 0-471-37407-5, California.

Press, W.H.; Teukolsky, S.A.; Vetterling, W.T. & Flannery, B.P.(1992) *Numerical Recipes in C: The Art of Scientific Computing 2nd edition*, Cambridge Univ. Press, ISBN : 0521437148,Cambridge U.K.

Ross A.; Dass, S. & Jain, A.(2006) Fingerprint Warping Using Ridge Curve Correspondences, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 28, Issue 1, Jan. 2006, pp. 19-30,ISSN:0162-8828.

Thai R. (2003) ,Fingerprint Image Enhancement and Minutiae Extraction, Honours Thesis, School of Computer Science & Software Engineering University of Western Australia,Australia

Tico,M. & Kuosmanen,P.(2000) An algorithm for fingerprint image postprocessing. *Proceedings of the Thirty-Fourth Asilomar Conference on Signals, Systems and Computers* , Vol.2,pp. 1735–1739, ISBN : 0-7803-6514-3,Pacific Grove CA USA, Nov. 2000.IEEE

Website of NEUROtechnology,Biomedical and Artificial Technologies, http://www.neurotechnologija.com/download.html

Yager, N & Amin, A. (2004). "Fingerprint classification: a review". *Springer-Verlag Pattern Analysis & Applications*, Vol. 7, Issue 1, April 2004,pp. 77–93,ISSN 1433-7541.

20

Multiple Self-Organizing Maps for Control of a Redundant Manipulator with Multiple Cameras

Nobuhiro Okada, Jinjun Qiu and Eiji Kondo
Kyushu University
Japan

1. Introduction

Vision guide for a manipulator has been one of the major research issues in robotics. Coordination schemes of visuo-motor systems can be classified on the basis of the knowledge about manipulator kinematics and camera parameters. Many researchers have proposed a number of systems that deal with unknown manipulator kinematics and unknown camera parameters. In the studies, visuo-motor models are either estimated analytically during the execution of tasks on-line or learned prior to the execution off-line. Artificial neural networks can be used to learn the non-linear relationships between features in images and the manipulator joint angles. Miller et al. proposed a neural network based on the learning control system, where a cerebeller model arithmetic computer memory was employed for the learning (Miller, 1989). Carusone et al. used a network to train an un-calibrated industrial robot (Carusone & Eleurterio, 1998). In their systems, neural networks provided the estimation of the poses of targets in the manipulator coordinate frames, and the poses were used to guide the manipulator to grasp the objects. However, supervisors were needed in the systems.
Self-organizing map (SOM) based on the Kohonen algorithm is an important unsupervised artificial neural network model (Kohonen, 1998). It has shown great potential in application fields such as motor control, pattern recognition, optimization, and so on, and also has provided insights into how mammalian brains are organized (Wiener et al., 2000) (Behera & Kirubanandan, 1999). During the past years it has been demonstrated that the SOM can solve the inverse kinematics problem for visuo-motor control. Buessler et al. determined arm movements by tracking an image target (Buessler & Urban, 1998) (Buessler et al., 1999). The correlation between an image-defined error and the joint movement was learned on-line using self-organizing algorithm for making the error zero. Multiple neural maps were combined to simplify neural learning in their study. Martinetz et al. and Walter et al. used a three dimensional lattice to learn the nonlinear transformation that specifies the joint angles of a 3-DOF manipulator so that the angles take the tip of the manipulator to a target point given in the coordinates provided by two cameras (Marinetz et al., 1990) (Walter & Schulten, 1993). In all of these studies, however, they solved the visuo-motor coordination problems

with non-redundant manipulators in an environment without obstacles. Such obstacle avoidance problems are important for manipulators that work in real environments. Zeller et al. developed a motion planning for a non-redundant manipulator to avoid collision with obstacles in a cluttered environment by using the TRN model (Zeller et al., 1997). They used a fact that a locally optimized path can be determined by minimizing the Euclidean distance from the current position to a given goal. Collision check was performed not in the self-organizing process but in the path planning process afterwards. Collobert developed a new organizing principle for perceptual systems based on multiple Kohonen self organizing maps. (Collobert, 2006) These maps are arranged in order to model the global brain activity as seen on tomography pictures. In contrast to these precedent studies, our system is not only for precise positioning of the end-effector but also for ensuring obstacle free poses of the manipulator using multiple SOMs. We intend to realize coordination for a visuo-motor system with a redundant manipulator in a cluttered environment. The redundancy is then used to make the manipulator take obstacle free poses and achieve high manipulability.

In the previous researches, Zha et al. used a SOM to coordinate a visuo-motor system in an environment with obstacles (Zha et al., 1996). Collisions between the links and obstacles were, however, not well considered. We introduced a potential field to avoid such collisions only in a 2D space (Okada et al., 1999). Han et al. realized collision avoidance for a visuo-motor system in a 3D space (Han et al., 2003). The occlusion problem was, however, not solved effectively even in the system.

Vision systems are generally classified by the number of cameras, camera configurations, the level of calibration and some a priori knowledge about the scene. The binocular configuration is a commonly used configuration. In comparison with the eye-in-hand configuration, it allows a wide field of view and then it makes easy to observe both the manipulator and targets simultaneously. Such a vision system was employed in Han et al.'s work. However, since they treated spaces occluded by obstacles in the image space as unreachable spaces for the manipulator, the workspace was restricted.

In order to handle the occlusion problem, we have developed a visuo-motor system with multiple related SOMs and a redundant camera system in this paper. The SOMs are directly connected to the cameras and learn to perform manipulator control. Based on the visibility of a target given in the workspace, the appropriate map is selected. The map outputs a joint angle vector which makes the manipulator reach the target with an obstacle free pose. The proposed learning algorithm ensures that the manipulator moves smoothly and consistently in the whole workspace no matter which map is selected. The advantages of the proposed method are: (1) By employing multiple maps, the system overcomes the occlusion problems in cluttered environments. The cooperation and complementation of maps make the manipulator consistently move in the whole workspace. (2) In our self-organizing learning procedure, the visuo-motor system learns not only to position the end-effector precisely but also ensure that the manipulator takes obstacle free poses.

2. Our Visuo-Motor System

Our visuo-motor system is illustrated in Fig.1. The system contains a 4-DOF redundant manipulator, multiple CCD cameras, and multiple related SOMs. The CCD cameras are used to get the target positions, the locations of the end-effector and the manipulator poses. They also acquire information about obstacles by using simple image technique. From visual

information provided by the cameras, the SOMs learn projections that convert the position vectors of the targets in the image spaces into the joint angle vectors of the manipulator.
Although stereo camera systems can provide 3D information and we have used such a system in our previous works, the system could not well deal with spaces occluded by obstacles. They introduced 3-cameras system to overcome the situation (Han et al., 2006). However, the result was limited. To deal with the occlusion problem, a multiple camera system is presented in this paper. The valid workspace is extended by using the cameras at multiple viewpoints. Related SOMs are simultaneously employed in the visuo-motor system.
Assume that the projections of a target point in the camera images are $u_{t\text{-}S1}$, $u_{t\text{-}S2}$, $u_{t\text{-}S3}$ (side camera1, 2, 3) and $u_{t\text{-}T}$ (top camera). A pair of image coordinates of the top camera and another side camera is combined into a 4-dimensional vector $u_{t\text{-}SOMn}$, and then it is used as an input to one of SOMs. Since the valid workspaces of the maps are different from each other, the maps are alternately used.

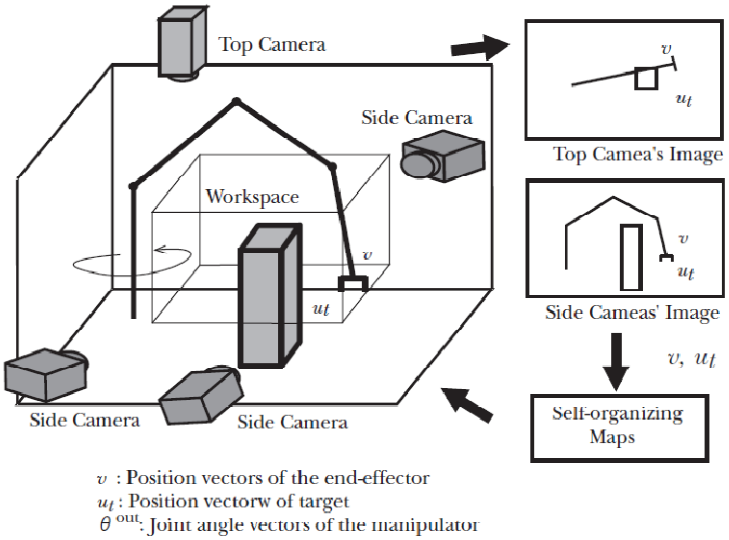


Fig. 1. Outline of our visuo-motor system

As shown in Fig.2, each SOM consists of neurons, which are distributed in 2 image spaces that correspond to cameras used as inputs. Each neuron has the following 4 parameters.
1. W: Position of the neuron in 2 image spaces.
2. J: Jacobi matrix from the manipulator joint angle space to the image spaces.
3. θ: Joint angle at W.
4. ξ: Gradient vector of the evaluation function H.
When a target u_t is given in the workspace, one of the maps is selected based on which camera can see the target. In the selected map, then, the neuron nearest to the projection of the target is chosen. The manipulator joint angle vector is finally calculated obeying the linear function (1).

$$\boldsymbol{\theta}_{out} = \frac{\sum g_n \left(\boldsymbol{\theta} + \mathbf{J}^{\dagger} \left(\mathbf{u}_t - \mathbf{W} \right) \right)}{\sum g_n} \tag{1}$$

Here J† is a pseudo-inverse matrix of J. Even though the projection from the image spaces to the joint angle space is not linear for a PUMA type manipulator, such a linear approximation can still be used in a small areas. In our SOMs, since many neurons are distributed in the image spaces, the area controlled by a neuron will be small enough.

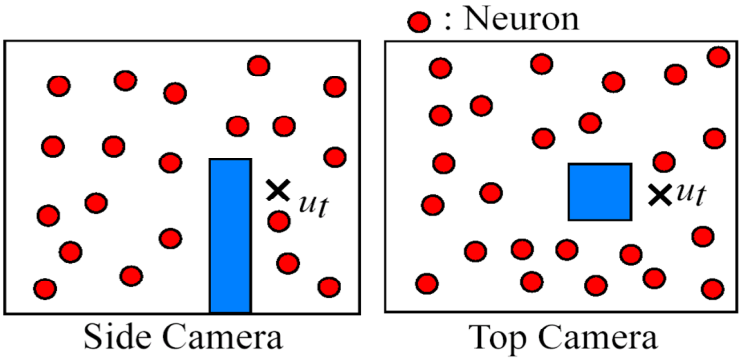


Fig. 2. A self-organizing map used in our system

In our actual system, weighted sum of outputs from multiple neurons around the target is used instead of (1). Where g_n is the weight defined by the following equation.

$$g_{nt} = \begin{cases} \exp(-n/\lambda) & \text{for } \exp(-n/\lambda) > \varepsilon \\ 0 & \text{for } \exp(-n/\lambda) \le \varepsilon \end{cases} \tag{2}$$

In the equation, n is the order of the neuron determined according to the distance between the neuron and the target. Nearer the target it has a larger value and farther it has a smaller value. The symbols λ and ε are values for defining the number of neurons that can affect θ_{out}.

3. Learning Procedure Of The Self-Organizing Maps

3.1 Reason for multiple SOMs

In our system, multiple SOMs are employed. If we utilize one SOM to corresponding to multiple cameras, the SOM will have a high dimensional space and be computationally difficultly. On the other hand, the distance between neurons in the SOM will be too huge to use a linear approximation. In addition, when a camera cannot see a target, the input of the SOM will become incomplete, therefore, the neurons cannot be all updated. To solve these problems, we utilize multiple SOMs.

If these multiple SOMs learn separately, therefore, outputs from them will be different with each other even for the same target in the workspace. This will result in that when the

system switches the outputs used for the manipulator control, the manipulator moves inconsistently. The problem should be effectively removed in the learning process. Then the problem can be described as: the learning algorithm has to guarantee that the manipulator moves smoothly and consistently in the whole workspace no matter which SOM is used for control.

Our learning procedure is explained using Fig.3. In the learning process, a target position u_t is randomly given in the workspace and the cameras obtain the position. At first we assume that the top camera (Fig.1) can always see the target in our former system. Each SOM is respectively related to a side camera and it receives target information from the side camera and the top camera (therefore the information is a 4-dimensional vector). Depending upon which side cameras can see the target, corresponding maps update their parameters. When only one side camera can see the target the corresponding SOM updates its parameter by itself. When more than 2 cameras can see it, the corresponding SOMs learn under influences from others. In the case, one of SOMs is arbitrary chosen to output the joint angles θ_{out}, and the manipulator is driven by it. Then all cameras that can see the end-effector obtain its position v. Finally each SOM related to the camera updates its parameters using u_t, θ_{out} and v. To remove the assumption that the top camera can always see the target, we increased the number of SOMs. The outputs from every 2 cameras form one SOM. The camera which can see the target is treated as the top camera of our former system. The SOMs learn using the learning algorithm motioned above. If more than 1 camera can see the target, the corresponding SOMs learn simultaneously. The updating algorithms will be described in the following subsections. The learning is done by iterating the above process for many targets.

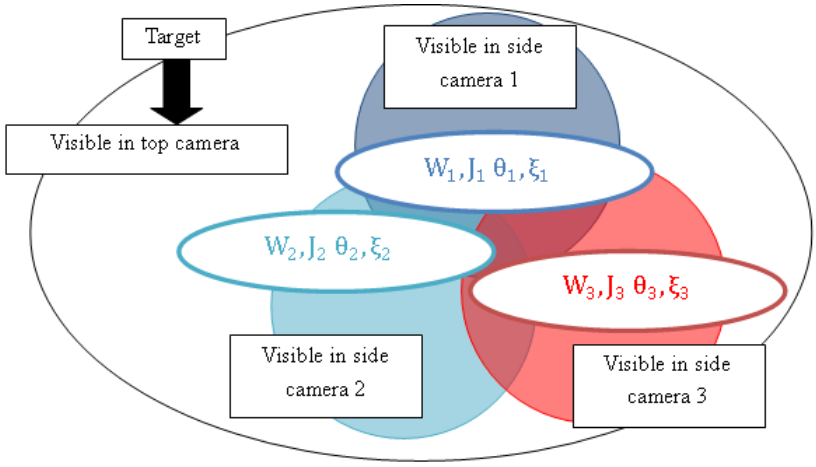


Fig. 3. Relation of multiple SOMs in the learning process

This learning procedure results in that the neurons of the multiple SOMs for a target possess similar values of θ_{out} in the end of learning even though other parameters may differ from each other. Thus the outputs from any SOMs will ensure the manipulator takes the same pose. By one of the major features of SOM, an assignment of similar joint angles to adjacent

targets is also achieved. Both features bring out a continuous and smooth transformation from the input image spaces of target positions to the output spaces of joint angles. Hence the SOMs guarantee smooth and consistent movements of the manipulator in the whole workspace.

3.2 SOM's learning procedure

A. Evaluation function

Calculating the inverse kinematics for a given end-effector position is a hard problem for redundant manipulators since it is an ill-posed problem that has many possible solutions. We introduced evaluation functions to resolve the under-determination into the system. In addition to making the end-effector reach the target position, the system outputs the joint angle configuration that optimizes the functions. One evaluation function is to achieve high manipulability, and the other is to make the manipulator take an obstacle free pose. The functions are respectively shown in (3) and (4). The total evaluation function is defined as the weighted sum of H_M and H_O (5). The function H_M is for manipulator with high manipulability, the other function H_O for manipulators with free obstacle poses.

$$H_M = \sqrt{\det\left(\mathbf{J}(\boldsymbol{\theta})\mathbf{J}^T(\boldsymbol{\theta})\right)} \tag{3}$$

$$H_O = \sum_l \left(1.0 - \frac{D_0}{d + D_0}\right) \tag{4}$$

$$H = \alpha_1 H_M + \alpha_2 H_O \tag{5}$$

Here d is the shortest distance from each link to the obstacles, and D_0 is the predefined value that provides the effective area of the potential. α_1 and α_2 are weights which are determined depending on the desirability of the individual functions.

B. Learning Algorithm

Initial neuron parameters of SOMs are randomly set at the beginning of the learning process. Then the joint angle outputs will lead the end-effector to wrong positions and make the manipulator take inadequate poses. By giving an arbitrary target in the workspace, they update the parameters. After many times iteration of the updating, an appropriate relation between the image spaces and the joint angle space will be established.

For N-th iteration, the SOMs updates their parameters using the learning flow mentioned below.

1) A target u_t is arbitrarily given in the workspace. The target positions in the camera images are extracted and transferred to SOMs.

2) Each SOM sorts its neurons in the ascending order of the distances between the target and W of the neurons.

$$\left\|\mathbf{u}_t - \mathbf{W}^1\right\| < \cdots < \left\|\mathbf{u}_t - \mathbf{W}^{\lambda^n}\right\| \tag{6}$$

Here λ^n is the number of neurons that will be updated in the n-th iteration.

3) One SOM is chosen and outputs the joint angles $\boldsymbol{\theta}_0^{out}$ by (7). The manipulator moves using $\boldsymbol{\theta}_0^{out}$.

$$\boldsymbol{\theta}_0^{out} = \frac{\sum_{i=1}^{\lambda^n} g\left(order_i^n, \lambda^n\right)\left(\boldsymbol{\theta}_i + \mathbf{J}_i^{\dagger}\left(\mathbf{u}_t - \mathbf{W}_i\right)\right)}{\sum_{i=1}^{\lambda^n} g\left(order_i^n, \lambda^n\right)} \tag{7}$$

Here $order_i^n$ is the order of i-th neuron. Since appropriate neuron parameters have not been obtained yet, however, the joint angles lead the end-effector to a wrong position. Then the cameras obtain the new end-effector position v_0 .

4) The SOM improves the output by (8) so that it can reduce the positioning error.

$$\boldsymbol{\theta}_1^{out} = \boldsymbol{\theta}_0^{out} + \frac{\sum_{i=1}^{\lambda^n} g\left(order_i^n, \lambda^n\right)\left(\boldsymbol{\theta}_i + \mathbf{J}_i^{\dagger}\left(\mathbf{u}_t - \mathbf{v}_0\right)\right)}{\sum_{i=1}^{\lambda^n} g\left(order_i^n, \lambda^n\right)} \tag{8}$$

The cameras obtain the new end-effector position v_1.

5) The SOMs update their neuron parameters as following subsection.

6) The system iterates the above process for defined times.

C. Updating the Parameters

Each neuron parameters are updated using u_t, θ_{out}, and v.

1) Updating W

W is updated by (9).

$$\mathbf{W}_i^{n+1} = \mathbf{W}_i^n + \varepsilon_W^n \, g\left(order_i^n, \lambda^n\right)\left(\mathbf{u}_t - \mathbf{W}_i^n\right) \tag{9}$$

Here ε_W^n is the learning coefficient for W. It has a large value for early stages of learning and has a small value for late stages. By updating W the neurons will be distributed all over the image spaces.

2) Updating J

J is updated by (10).

$$\mathbf{J}_i^{n+1} = \mathbf{J}_i^n + \varepsilon_J^n \, g\left(order_i^n, \lambda^n\right)\Delta\mathbf{J}_i^n \tag{10}$$

Here ε_J^n is the learning coefficient for J, and it changes just like ε_W . $\Delta\mathbf{J}_i^n$ is determined by Widrow-Hoff's learning rule. The error function E_J is

$$E_J = \frac{1}{2}\left\|\left(\mathbf{v}_1 - \mathbf{v}_0\right) - \mathbf{J}_i^n\left(\boldsymbol{\theta}_1^{out} - \boldsymbol{\theta}_0^{out}\right)\right\|^2 \tag{11}$$

Then $\Delta\mathbf{J}_i^n$ is determined by

$$\Delta\mathbf{J}_i^n = -C_J\frac{\partial E_J}{\partial \mathbf{J}_i^n} = \frac{\left(\mathbf{v}_{01} - \mathbf{J}_i^n\,\boldsymbol{\theta}_{01}^{out}\right)\boldsymbol{\theta}_{01}^{out\,T}}{\left\|\boldsymbol{\theta}_{01}^{out}\right\|^2} \tag{12}$$

Here $\mathbf{v}_{01} = \mathbf{v}_1 - \mathbf{v}_0$, $\boldsymbol{\theta}_{01}^{out} = \boldsymbol{\theta}_1^{out} - \boldsymbol{\theta}_0^{out}$, and $C_J = 1/\left\|\boldsymbol{\theta}_{01}^{out}\right\|^2$. By updating J SOMs become to output more appropriate θ_{out}.

3) Updating ξ

ξ is the gradient vector of the evaluation function.

$$\boldsymbol{\xi} = \alpha_M\boldsymbol{\xi}_M + \alpha_O\boldsymbol{\xi}_O \tag{13}$$

Here ξ_M is the gradient vector of H_M, ξ_O is the gradient vector of H_O.

ξ_M is updated by (14).

$$\boldsymbol{\xi}_{M,i}^{n+1} = \boldsymbol{\xi}_{M,i}^n + \varepsilon_{\xi M}^n\, g\left(order_i^n, \lambda^n\right)\Delta\boldsymbol{\xi}_{M,i}^n \tag{14}$$

Here $\varepsilon_{\xi M}^n$ is the learning coefficient similar to ε_W^n . $\Delta\boldsymbol{\xi}_{M,i}^n$ is determined likely with $\Delta\mathbf{J}_i^n$. The error function is

$$E_{\xi M} = \frac{1}{2}\left\|\left(H_{M,k} - H_{M,j}\right) - \boldsymbol{\xi}_{M,i}^{n\,T}\left(\boldsymbol{\theta}_k^{out} - \boldsymbol{\theta}_j^{out}\right)\right\|^2 \tag{15}$$

Here j is the ID number of the neuron that is the closest to the target, and k is the ID number of the 2nd neuron. Then $\Delta\boldsymbol{\xi}_{M,i}^n$ becomes as following.

$$\Delta\boldsymbol{\xi}_{M,i}^n = -C_{\xi M}\frac{\partial E_{\xi M}}{\partial \boldsymbol{\xi}_{M,i}^n} = \frac{\left(H_{M,jk} - \boldsymbol{\xi}_{M,i}^{n\,T}\,\boldsymbol{\theta}_{jk}^n\right)\boldsymbol{\theta}_{jk}^{n\,T}}{\left\|\boldsymbol{\theta}_{jk}^n\right\|^2} \tag{16}$$

Here $H_{M,jk} = H_{M,k} - H_{M,j}$, $\boldsymbol{\theta}_{jk}^{n} = \boldsymbol{\theta}_{k}^{n} - \boldsymbol{\theta}_{j}^{n}$, and $C_{\xi M} = 1/\left\|\boldsymbol{\theta}_{jk}^{out}\right\|^2$.
ξ_O is updated by (17).

$$\boldsymbol{\xi}_{O,i}^{n+1} = \boldsymbol{\xi}_{O,i}^{n} + \varepsilon_{\xi O}^{n}\, g\left(order_i^n, \lambda^n\right)\Delta\boldsymbol{\xi}_{O,i}^{n} \tag{17}$$

Here $\varepsilon_{\xi O}^{n}$ is the learning coefficient similar to ε_{W}^{n}. $\Delta\boldsymbol{\xi}_{O,i}^{n}$ is also determined likely with $\Delta\mathbf{J}_i^n$. The error function is

$$E_{\xi O} = \frac{1}{2}\left\|\left(H_{O,1} - H_{O,0}\right) - \boldsymbol{\xi}_{O,i}^{n\ T}\left(\boldsymbol{\theta}_1^{out} - \boldsymbol{\theta}_0^{out}\right)\right\|^2 \tag{18}$$

Here $H_{O,0}$ and $H_{O,1}$ are respectively the potential values for v_0 and v_1. Then $\Delta\boldsymbol{\xi}_{O,i}^{n}$ becomes as following.

$$\Delta\boldsymbol{\xi}_{O,i}^{n} = -C_{\xi O}\frac{\partial E_{\xi O}}{\partial \boldsymbol{\xi}_{O,i}^{n}} = \frac{\left(H_{O,01} - \boldsymbol{\xi}_{O,i}^{n\ T}\boldsymbol{\theta}_{01}^{n}\right)\boldsymbol{\theta}_{01}^{n\ T}}{\left\|\boldsymbol{\theta}_{01}^{n}\right\|^2} \tag{19}$$

Here $H_{O,01} = H_{O,1} - H_{O,0}$, $\boldsymbol{\theta}_{01}^{n} = \boldsymbol{\theta}_{1}^{n} - \boldsymbol{\theta}_{0}^{n}$, and $C_{\xi O} = 1/\left\|\boldsymbol{\theta}_{01}^{out}\right\|^2$.
By updating ξ using (14) and (17), the SOMs become to output joint angles that make the manipulator achieve high manipulability and take obstacle free poses.
4) Updating θ
θ is updated by (20).

$$\boldsymbol{\theta}_i^{n+1} = \boldsymbol{\theta}_i^{n} + \varepsilon_{\theta}^{n}\, g\left(order_i^n, \lambda^n\right)\Delta\boldsymbol{\theta}_i^{n} \tag{20}$$

Here ε_{θ}^{n} is the learning coefficient similar to ε_{W}^{n}. $\Delta\boldsymbol{\theta}_i^{n}$ is determined as

$$\Delta\boldsymbol{\theta}_i^{n} = \boldsymbol{\theta}_i^{Desire} - \boldsymbol{\theta}_i^{n} \tag{21}$$

$$\boldsymbol{\theta}_i^{Desire} - \boldsymbol{\theta}_0^{out} = \mathbf{J}_i^{n\dagger}\left(\mathbf{W}_i^{n} - \mathbf{v}_0\right) + \left(\mathbf{I} - \mathbf{J}_i^{n\dagger}\mathbf{J}_i^{n}\right)\boldsymbol{\xi}_i^{n} K_p^{n} \tag{22}$$

Here K_p^n is a positive coefficient, it is introduced to realize high manipulability and obstacle free poses. It also decreases with learning times in order to enable fine tuning of the system. Then $\Delta\boldsymbol{\theta}_i^{n}$ becomes as following.

$$\Delta\boldsymbol{\theta}_i^{n} = \boldsymbol{\theta}_i^{out} - \boldsymbol{\theta}_i^{n} + \mathbf{J}_i^{n\dagger}\left(\mathbf{W}_i^{n} - \mathbf{v}_0\right) + \left(\mathbf{I} - \mathbf{J}_i^{n\dagger}\mathbf{J}_i^{n}\right)\boldsymbol{\xi}_i^{n} K_p^{n} \tag{23}$$

By updating θ using (20), the SOMs become to output θ^{out} so that the manipulator moves its hand-effector with less error, that it achieves high manipulability, and that it takes obstacle free poses.

4. Path Planning

Collision avoidance in our system is realized on the basis of the following idea. While the path projections in any one camera images do not interfere with the obstacle projections, the path does not collide with the obstacle in 3-dimensional space. Also the SOMs determine joint angles of the manipulator so that it takes obstacle free poses for given target position. By combining the SOMs and a simple path planning system, then, we can realize collision avoidance. The path planning system only has to make a trajectory that does not collide with obstacles in the camera images. To control the manipulator by the SOMs outputs for points on the trajectory is to realize obstacle avoidance. The idea is different from most of existing algorithms that uses configuration space. Our planning system plans paths in 2-dimensional spaces and then the computational cost is lower than them. The system adopts Laplace potential method. The method can avoid local minima. For detail, please refer to our previous study (Han et al., 2006).

5. Simulation Results

We have constructed an experimental system. The outline of the system is shown in Fig.4. By simulation and experiments, we have also revealed that a visuo-motor system with 3 CCD cameras and 2 SOMs can control a redundant manipulator and realize collision avoidance in an environment with obstacles.

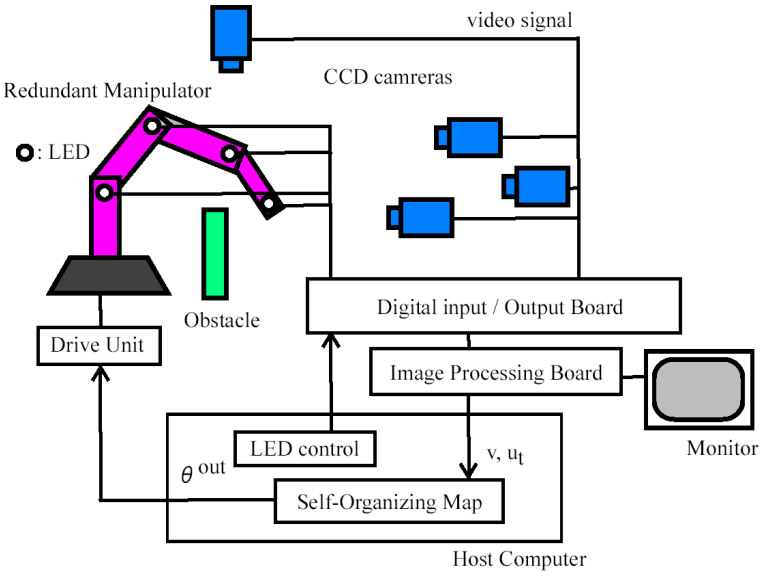


Fig. 4. Outline of our experimental system

In this paper, we aim at a system with 4 CCD cameras and a 4-DOF redundant manipulator, and show its validity by simulation. The cameras are assumed to be orthographic models and each camera has 640*480 pixel resolution. The simulation model is illustrated in Fig. 5. The length of each link is 120, 135, 110 and 140 pixels in the image spaces. Each SOM involves 240 neurons. 15000 targets were given in the learning process and they were distributed within 200*200 pixel rage with a focus on the obstacle. The time required for the learning was about 10 minutes using a PC with 3.0GHz Pentium4. The learning parameters are listed in Table 1.

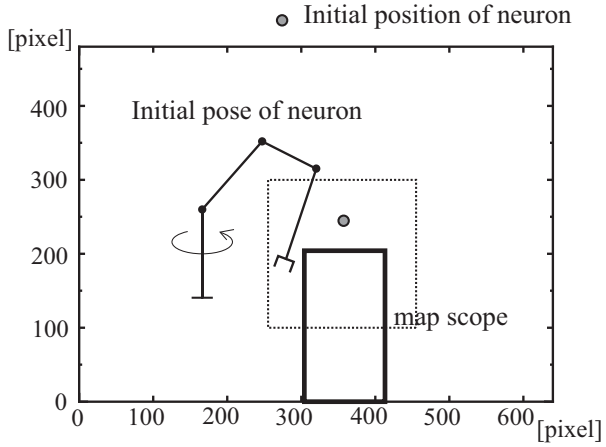


Fig. 5. Assumed simulation environment illustrated in the image space of the right camera

After the learning, 45 target positions were given randomly to evaluate the positioning error of the end-effector and to confirm the poses of the manipulator. The following figures show the positions of the end-effector and poses of the manipulator obtained by each camera. "x" marks are positions of the end-effector, and "o" are the projected targets.

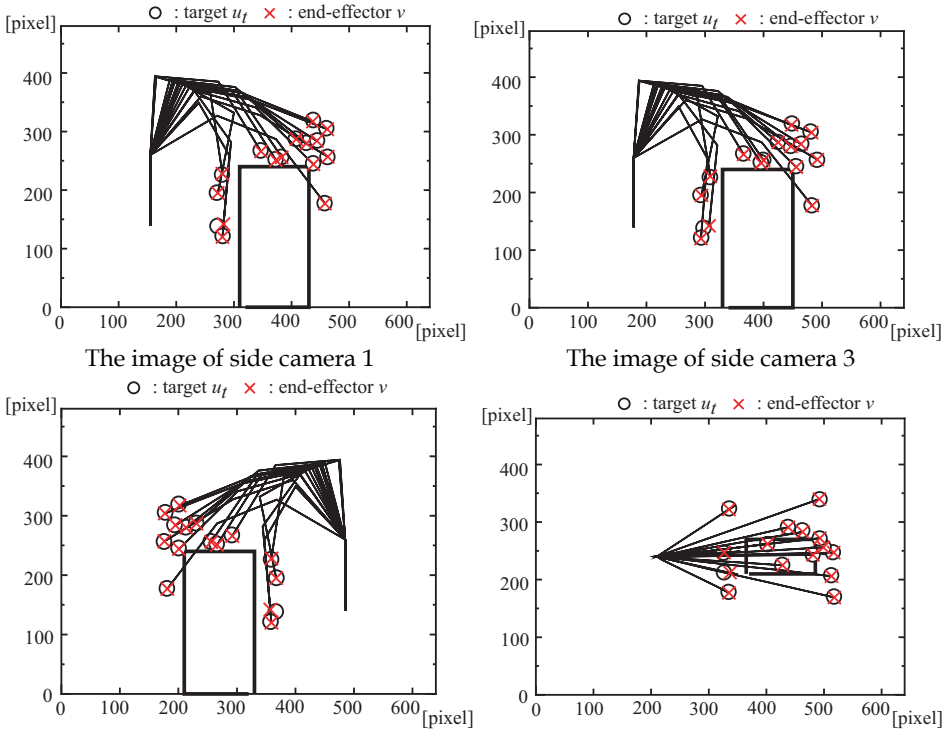
	λ	ξ_W	ξ_O	K_p	α_1	α_2
Initial value	48	1.0	0.3	0.24	5.0	0.3
Final value	1	1.0	1.0	0.0012		

Table 1. Learning parameters used in the simulation

Fig.6 shows targets that are visible from all cameras and poses of the manipulator. It can be seen that there are some poses that touch the obstacle in the image spaces of side cameras. However, the manipulator does not collide with the obstacle with the poses indeed, since they are obstacle-free poses in the image space of the top camera. The figures also show that each SOM keeps continuousness with each other. The average positioning error of the end-effector was 1.89 pixels.

When both the initial and the goal positions can be observed in one side camera, the collision avoidance can be performed by using the only the corresponding SOM. An example of path planning is shown in Fig.7. The path planning system planned a collision-free path of the end-effector in the top camera image using Laplace potential method, and

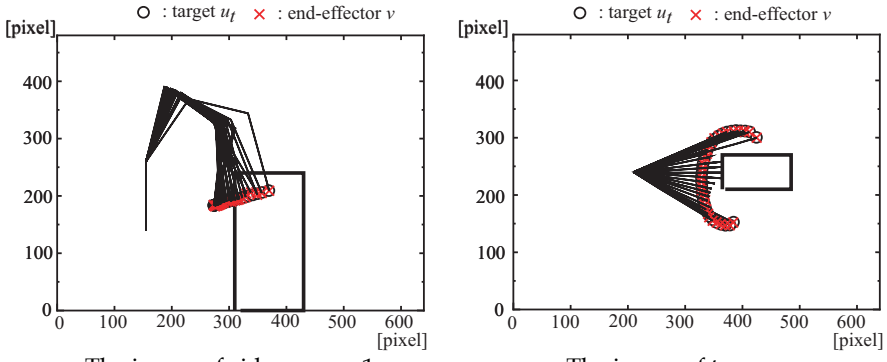
determined the shortest path in another camera image. Then the planning system divided the path into 34 positions and the SOMs outputted the collision-free poses for the positions. The figures show that the planned path avoided the obstacle. Also, they show that the system controlled the manipulator switching outputs from the SOMs in an environment where occlusion occurred.



The image of side camera 1 The image of side camera 3

The image of side camera 3 The image of top camera

Fig. 6. Targets and manipulator poses after the learning process



The image of side camera 1 The image of top camera

Fig. 7. Collision avoidance by the simulation

6. Conclusions

We developed a visuo-motor system with multiple self-organizing maps. The system consists of a redundant manipulator, multiple cameras and multiple SOMs corresponding to the cameras. By using the cameras, the system can control the manipulator in an environment with obstacles. To realize the system, we also developed a learning method of the SOMs. The learning method can keep consistency of outputs among the SOMs. We then combined the SOMs and a path planning system to achieve collision avoidance. Since the SOMs outputs collision free poses, the path planning system became very simple. Simulation results showed that the SOMs control the manipulator with obstacle free poses and that the collision avoidance was realized.

7. References

Behera, L. & Kirubanandan, N. (1999). A hybrid neural control scheme for visuo-motor coordination, *IEEE Control Systems*, pp.34-41.

Buessler, J. L.; Kara, R. ; Wira, P. ; Hihl, H. & Urban, J. P. (1999). Multiple self-organizing maps to facilitate the learning of visuo-motor correlations, *Proc. of the IEEE Int. Conf. on Systems, Man and Cybernetics*, pp.470-475.

Buessler, J. L. & Urban, J. P. (1998). Visual guided movements: learning with modular neural maps in robotics, *Neural Networks*, 11, pp.1395-1415.

Carusone, J. & Eleurterio, G. (1998). The feature CMAC: a neural-network based vision system for robotic control, *Proc. of the IEEE Int. Conf. on Robotics and Automation*, vol.4, pp.2959-2964.

Collobert, M. (2006). Using multi-kohonen self-organizing maps for modeling visual perception, *Computer Vision and Graphics*, pp.1031-1036.

Han, M.; Okada, N. & Kondo, E. (2003). Collision avoidance for a visuo-motor system using a self-organizing map in a 3D space, *6th Japan-France Congress on Mechatronics*, pp.495-500.

Han, M. ; Okada, N. & Kondo, E. (2006). Coordination of an uncalibrated 3-D visuo-motor system based on multiple self-organizing maps, *JSME International Journal*, Series C, vol.49, pp.230-239.

Kohonen, T. (1998). Self-organizing maps and associative memory, *Springer information Sciences*, vol.8, pp.43-48.

Marinetz, T. ; Ritter, H. & Schulten, K. (1990). Three-dimensional neural net for learning visuo-motor coordination of a robot arm, *IEEE Trans. on neural networks*, vol.1, pp.131-136.

Miller, W. (1989). Real-time application of neural networks for sensor-based control of robots with vision, *IEEE Trans. on Systems, Man and Cybernetics*, vol.19, No.4, pp.825-831.

Okada, N. ; Shimizu, Y. ; Maruki, Y. & Yoshida, A. (1999). A self-organizing visuo-motor map for a redundant manipulator in environment with obstacles, *Proc. of the 9th Int. Conf. on Advanced Robotics*, pp.517-522.

Walter, J. A. & Schulten, K. J. (1993). Implementation of self-organizing neural networks for visuo-motor control of an industrial robot, *IEEE Trans. on Neural Networks*, vol.4, pp.86-95.

Wiener, J.; Burwick, T. & von Seelen, W. (2000). Self-organizing maps for visual feature representation based on natural binocular stimuli, *Biological Cybernetics*, 82, pp.97-110.

Zeller, M. ; Sharma, R. & Schulten, K. (1997). Motion planning of a pneumatic robot using a neural network, *IEEE Control Systems*, 17, pp.89-98.

Zha,H. B.; Onitsuka, T. & Nagata,T. (1996). A visuo-motor coordination algorithm for controlling arm's movement in environments with obstacles, *Proc. of the 4th Int. Conf. on Control, Automation, Robotics and Vision*, pp.1013-1017.

21

Tracking English and Translated Arabic News using GHSOM

Ali Selamat[α] and Hanadi Hassen Ismail Mohammed[β]
[α]*Universiti Teknologi Malaysia, Malaysia*
[β]*Sudan University of Science and Technology, Sudan*

1. Introduction

After the September eleventh attacks, the media has contributed in clarifying the discrepancies between eastern and western cultures. As stated by Michael Binyon, a journalist in the The Times newspaper, whether they work for newspapers, television, or radio programs, all journalists are bound by their professional ethics codes, to provide truthful and accurate news. Unfortunately, only very few media outlets can boast of saying that everything they have put out is everything that had really happened (Michael Binyon, 2008). Take, for example, the violence in Gaza, a journalist was asked by the publisher of a Middle Eastern newspaper why the BBC gave so much coverage to the rockets aimed by Hamas at Israeli towns but gave so little coverage to the Israeli air strikes in Gaza. His answer was because there were no BBC correspondents based in Gaza. This shows that there is a need to track similar news from different resources because watching news from one resource may not always give audience the real picture of the event.

There are many advantages of finding similar content across different languages. For example, it can form the basis for multilingual summarization and the question answering support for web pages that provide questions and answers. It also facilitates comparative studies across national, ethnic, and cultural groups (Jin & Barrire 2005). Dittenbach, et al.,2001 have stated that human categorizations are based on grouping similar objects into a number of categories. This ahs been done in order to understand the differences between objects that belong to each defined category. There are many approaches have been used for finding similarity across multilingual documents such as neural networks, fuzzy clustering, genetic algorithms, support vector machines, etc. Most of these techniques are baed on the assumption on the availability of a clean corpus and the majority of these pages are written independently of each other (Jin & Barrire 2005). Therefore, two versions of the same topic that are written in two different languages cannot simply be taken as parallel corpora. One of the techniques used to solve this problem is Self-Organizing Map (SOM) (Kohonen, 1990). It is an unsupervised neural network algorithms which provide a topology-preserving mapping from a high-dimensional document space into a two dimensional map space. The documents that are on similar topics are located in neighboring regions (Dittenbach, et al.,2001). SOM produces additional information about the affinity or similarity between the clusters themselves by arranging them on a 2D rectangular or

hexagonal grid when we cluster the documents. Similar clusters are neighbors in the grid, and dissimilar clusters are placed far apart in the grid (Zamir, 1999).
Similarity techniques have been used to find important information in the summarization of English news but it has not been used across languages (Chen et al., 2004). Finding similar news documents in Arabic and English news using machine learning algorithms is not an easy task. There are many news agencies nowadays that are broadcasting on what is happening around us and around the world such as Aljazeera (Aljazeera, 2007), Alsharq Alawsat (Alsharqalawsat, 2007), CNN (CNN, 2007), BBC (BBC, 2007), etc. Large networks provide more national and global news, while local stations concentrate more on the regional issues. Furthermore, even though everyone in the news industry claims their reporting is objective, the actual attitude in the broadcasting may be biased and different from network to network due to the background and cultural differences. Therefore, getting the same news from multiple sources provides the audience with more comprehensive views. These views would also be used by intelligence analyst in assessing counter-terrorism, political leadership, or country specific information.
In this paper, we analyze the appropriate techniques to find similar news across Arabic and English sources. This will provide the audience with multiple views of the broadcasted news because reading the news from a single source may not always reflect what is happening around the world due to different background, cultures, and opinions of the readers and writers. We have analyzed the similarity of the views on the news written in the news translations form Arabic and English texts using Self-organizing Map (SOM) (Kohonen, 1990). However, we have found that there are some difficulties in SOM that affect its performance. In order to improve its performance, we have used a Growing Hierarchical Self-Organizing Map (GHSOM) (Helmut, Dieter 2005). The research that has been undertaken is described in the rest of the sections.

2. Related Research On Arabic and English Translations

The discovering of new knowledge from textual information sources has increased the popularity in the information systems field. This is particularly important when an increasing availability of digital documents in various languages has enabled the web documents to be accessed by many internet users (Lee & Yang 2003).
To cross language boundaries between different languages, dictionaries are the most typical tools. However, the general-purpose dictionary is less sensitive in genre and domain and it is impractical to manually construct multilingual dictionaries for large applications. Corpus-based approaches, which do not have the limitation of dictionaries, provide a statistical translation model to cross the language boundary. Parallel corpora form the basis of much multilingual research in natural language processing (Lee & Yang 2003). There are different approaches for finding similar sentences across multiple languages. Most methods for finding similar sentences assume the availability of a clean parallel corpus. In some web pages that provide multiple languages, two versions of a page topic in two different languages are a good starting point for searching similar sentences. However, these pages may not always conform to the typical definitions of a bitext which current techniques assume.

Bitext generally refers to two versions of a text in two different languages (Adafre & Rijke, 2006). However, it is not known how the information is shared among the different languages in the same web page. Some pages tend to be the translations of each other. Thus, it is easy to use the automatic Parallel Text Identification (PTI) systems for aligning parallel web documents, which are direct translations of each other. The PTI systems crawl the Web to fetch potentially parallel multilingual Web documents (e.g Web spider) and to determine the parallelism between potential document pairs. Two modules are developed here. First, a filename comparison module is used to check the filename resemblance. Second, a content analysis module is used to measure the semantic similarity. The multilingual web documents retrieved by the web spider are initially passed to the filename comparison module to undergo a filename comparison process. Any two web pages in two different languages with their corresponding filenames resemble each other are picked up and aligned to form a parallel document pair. The multilingual web documents that remain unaligned after the filename comparison process will be passed to the content analysis module to carry out semantic content analysis (Chen et al., 2004).

3. Finding News Similarity Using SOM and GHSOM

3.1 News Collection

The available news documents were collected from several news websites such as Aljazeera news (Aljazeera, 2007), Alsharq Alawsat (Alsharqalawsat, 2007), etc. We have used the news crawler software (Selamat A.et al., 2007), available from the internet in order to get the desired URLs and then download the news materials. The Arabic news collection consists of the translated version of the news.

3.2 Documents Preprocessing

Documents preprocessing refers to the act of cleaning the text and processing the data before it is parsed. It consists of three main operations. Firstly, the conversion of HTML files to plain text files, where the downloaded HTML files were striped from all HTML-tags, and then converted to plain text files using the available public HTML-to-text converters. Secondly, the removal of stop words from the documents, where all the words which are not important to retrieval will be removed because they exist in any document repeatedly and do not affect the meaning. In order to remove the stop words, a file containing a large number of common stop words is used.

The news text words are matched with the words in the stop words list. If this matches, the word is automatically excluded from the text. Thirdly, the stemming operation, where prefixes and suffixes will be removed to obtain word stems. This can be achieved using a stemming program. The stemming is highly language-dependent. For the English language, the well-known and well-established Porter's stemmer (Selamat A. & Omatu S, 2004) provides high quality stemming for content representation purposes. The translated Arabic news contents will also be stemmed using the same stemmer used for the English content.

3.3 Documents Parsing

Parsing is the process of creating a vector representation of the texts that can be used for training a self-organizing map. A list of all words occurring in the document collection is called the template vector. To obtain the individual vectors, every document is described by the words occurring in it. In the simplest form of document representation, a binary indication is used for describing the fact whether a word is present or not, leading to a corresponding vector representation of 0 or 1. This type of representation is usually referred to as binary representation. The importance of every word in each document is based on the term frequencies that will be obtained using the so-called term frequency time's inverse document frequency (tf x idf) representation, where the importance of each word is judged with respect to the number of documents it appears in, as appointed in chapter two.

3.4 Creating feature vectors

A numerical representation of the documents collection was created in order to be able to automatically organize documents by content. This representation is created by full text indexing the documents. The template vector is the list of all words occurring in the document collection. This template vector consists of a very huge number of words. For better representation, this list should be pruned to a subset relevant and this can be achieved by removing all words that appear in a very large number (e.g., more than 90%) of documents, as these words do not contribute to content separation. This process also includes removing the stop words such as *a*, *the*, *you*, *we*, and *they*. The advantages of this process are the non-significant words are removed and the total size of the document file is reduced. Words that appear in only very few documents were also be removed.

3.5 Growing Hierarchical Self-organizing Maps

The Growing Hierarchical Self-Organizing Map (GHSOM), which is an extension to the growing grid SOM and hierarchical SOM, has been used to build a hierarchy of multiple layers where each layer consists of several independent growing SOMs. The size of these SOMs and the depth of the hierarchy are determined during its learning process according to the requirements of the input data. For the initial setup of the GHSOM, at Layer 0, a single-neuron SOM is created and the neuron's weight vector is initialized as the average of all the input vectors. Then, the learning process starts at Layer 1 with a small SOM (usually a grid) whose weight vectors are initialized to random values. The GHSOM grows in two dimensions: horizontally (by increasing the size of each SOM) and hierarchically (by increasing the number of layers).

For the horizontal growth, shown in Fig. 1, each SOM modifies itself in a systematic way, very similar to the growing grid so that each neuron does not represent too large of an input space. For the hierarchical growth, shown in Fig. 2, the principle is to periodically check whether the lowest layer SOMs have achieved sufficient coverage for the underlying input data (Tangsripairoj, Samadzadeh 2005). The basic steps of the horizontal growth and the hierarchical growth of the GHSOM are summarized according to Tangsripairoj et al.(Tangsripairoj, Samadzadeh 2005).

The growth process of the GHSOM is controlled by the following four important factors. First, the quantization error of a neuron ^{i,qe_i} is calculated as the sum of the distance between the weight vector of neuron i and the input vectors mapped onto this neuron.

Second, the mean quantization error of the map (mqe_m) is the mean of all neurons quantization errors in the map. Third, the threshold τ_1 is for specifying the desired level of detail that is to be shown in a particular SOM. Fourth, the threshold τ_2 is for specifying the desired quality of input data representation at the end of the learning process.

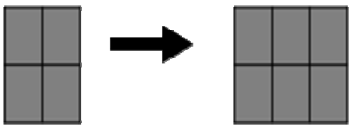


Fig. 1. GHSOM addition of new column

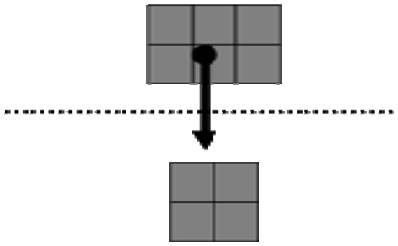


Fig. 2. GHSOM expansion in a sub layer

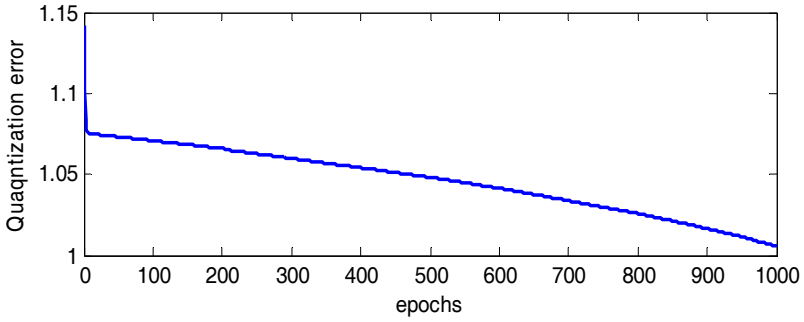


Fig. 3. Quantization Error for each epch

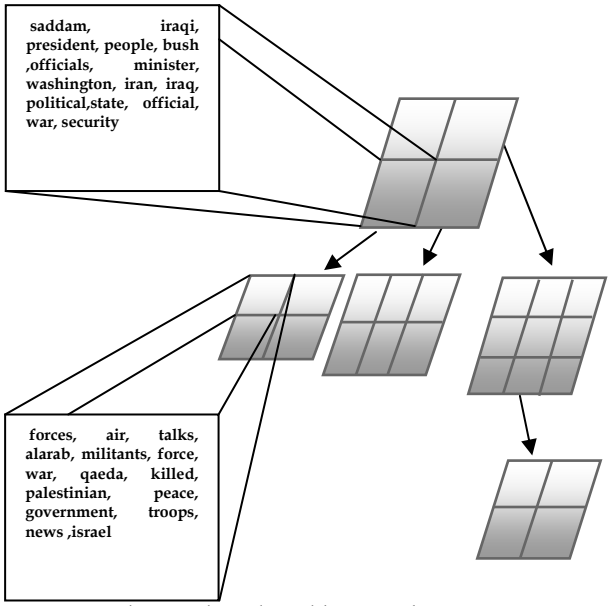


Fig. 4. 3-Layer GHSOM map with sample political keywords

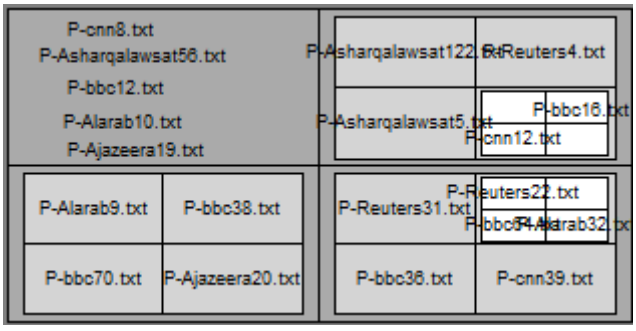


Fig. 5. 3-Layer GHSOM map with political documents cluster

4. Discussion of Results

4.1 Map quality

For exploratory data analysis tasks where the SOM and GHSOM are used as tools to display similarity relationships from a high-dimensional input space on a low dimensional mapping space, the quality of this mapping is essential. The quality of data representation is measured in terms of the deviation between a units model vector Best Matching Unit (BMU) and the subset of input data vectors represented by this unit. The deviation can be calculated as either the mean quantization error (*mqe*) or the quantization error (*qe*) of the single unit. This error is used to measure the map resolution as mentioned by Liu et al. (Liu, et al., 2005).

$$\left[mqe_i = \frac{1}{n} \bullet \sum_j \left\| m_i - x_j \right\| \right] \quad (1)$$

Where m_i is the codebook vector for *unit i* , x_j for all *j* are the input vectors mapped on *unit i*, and n is the number of *xi*. Referring to Fig. 5, using different vector size, the GHSOM achieves better results or less quantization error in several vector sizes than SOM. This means that GHSOM has the ability to make better map representation than SOM.

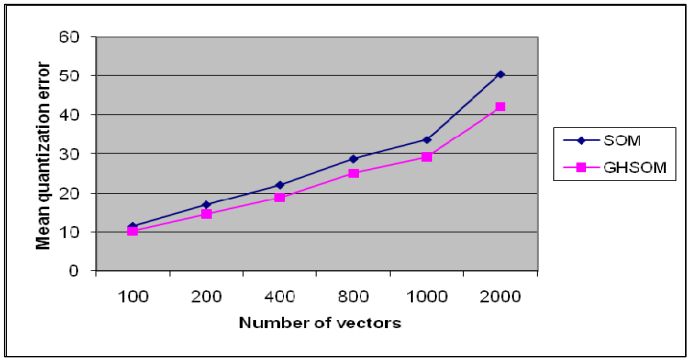


Fig. 6. Mean Quantization Error for SOM and GHSOM.

4.2 Documents Similarities Measurements

In the context of document clustering, the ability of an algorithm to classify a document into one or several categories is of high interest to the user. The classification effectiveness of this study has been measured using standard information retrieval measurements that are Precision(P), Recall(R), and F1. These can be expressed as:

$$\Pr ecision = \frac{a}{a+b} \quad (2)$$

$$\mathrm{Re}\, call = \frac{a}{a+c} \quad (3)$$

$$F1 = \frac{2PR}{P+R} \quad (4)$$

Category set $C = c_1, c_2, \ldots, cn$		Expert Judgement	
		Yes	No
System Judgement	Yes	a	b
	No	c	d

Table 1. Exploration of Measurements Used In the Experiments

Four main categories have been used to train the networks that are business, politics, sports, and technology. The Precision, Recall, and F1 measures of using SOM and GHSOM that have been calculated for all of these categories are shown in Table2, Fig. 6,7, and 8. Plots for these categories in Fig. 6 and 7 show that the GHSOM algorithm has consistently achieved higher Precision and Recalls levels than the standard SOM. Referring to Fig. 6 and 7, we have found that the average of both the Precision and Recall measures for SOM and GHSOM are 87% and 93%, respectively.

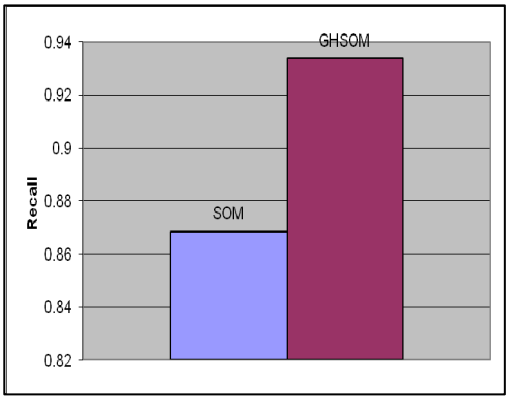


Fig. 7. Average Precision for SOM and GHSOM.

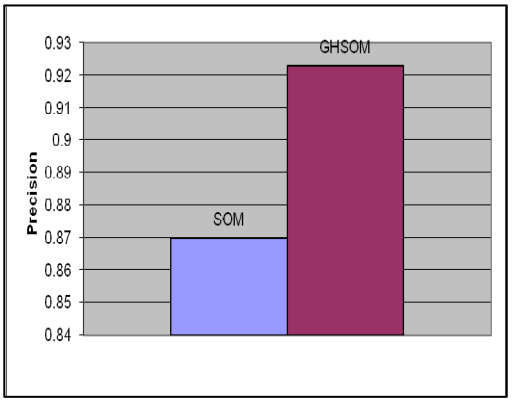


Fig. 8. Average Recall for SOM and GHSOM.

Figure 8 shows the achievable F1 measures for all categories. It is defined as the harmonic mean of Precision and Recall as shown in Eqs. (2), (3), and (4), respectively, and the yields values in the interval [0, 1], with F1 = 0 when no relevant documents are found, and F1 = 1 when all documents from a given class are retrieved with no errors. The detail explanation on each category of document similarity is shown in Table2. As shown in the table, the GHSOM outperforms the original SOM algorithm in all clusters except at 50 documents when the precision is the same in Business and Sports clusters. Referring to Fig. 8, we have

found that the average of F1 measures for SOM and GHSOM are 87% and 92%, respectively. It shows that the GHSOM algorithm has performed better in all categories.

Precision	Business		Politics		Sports		Technology	
	SOM	GHSOM	SOM	GHSOM	SOM	GHSOM	SOM	GHSOM
At 50 docs	1	1	0.88	1	1	1	0.75	1
At 100 Docs	0.96	1	0.88	1	0.88	1	0.82	1
At 500 Docs	0.94	0.96	0.89	0.98	0.95	0.97	0.59	0.83
At 1000 Docs	0.90	0.96	0.89	0.94	0.98	0.93	0.70	0.87

Table 2. Precision at 1000 documents of four clusters.

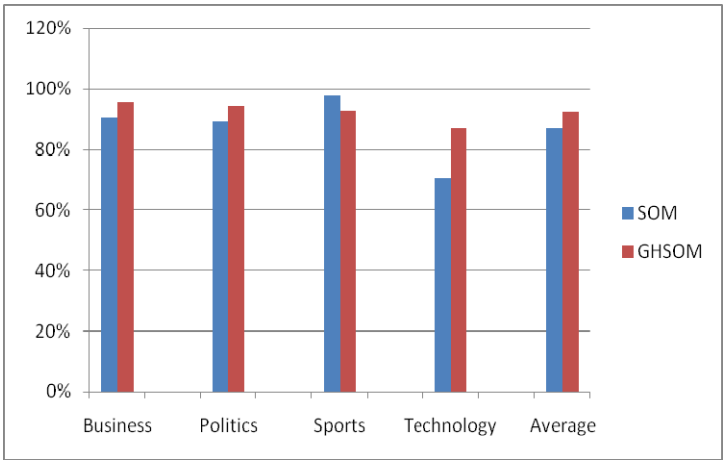


Fig. 9. F1 measures for four categories achieved by SOM and GHSOM.

Average of F1 measures for SOM and GHSOM are 87% and 92%, respectively.

4.3 The resulting map

The SOM and GHSOM maps consist of cells where each cell contains similar content documents, as shown in Fig. 4. The keywords appear in the figure indicate that these key keywords have higher frequency. Thus, most of the documents containing these keywords will be mapped in these cells, as shown in Fig. 5. In order to compare the results of the SOM and GHSOM applications, both algorithms were trained using large set of documents (1000 documents). Because of the limited space, we do not show the full maps that contain all the documents. As shown in Fig. 10, the SOM map contains the four main categories that are business, politics, sports, and technology as labeled with cluster titles. News with similar features are apparently located on the nearby regions of the map. The problem with the SOM map is that the map size should be predetermined and whenever the input space becomes larger, it is getting difficult to clearly visualize the map and the clusters areas. In the GHSOM resulted map illustrated in Fig. 11, it can be seen that the clusters are the areas with high densities on the map that were further hierarchically expanded by the growing

SOMs. In the figure, the top layer maps are depicted in gray and the bottom layer maps are depicted in white.

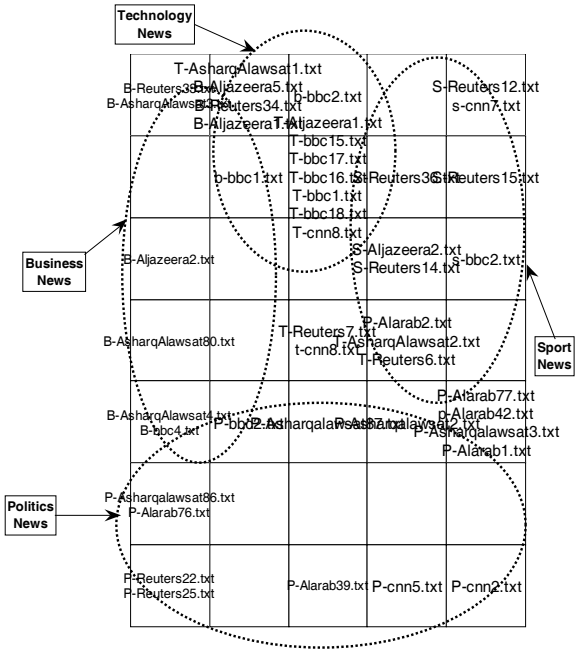


Fig. 10. The resulting 7X5 SOM map.

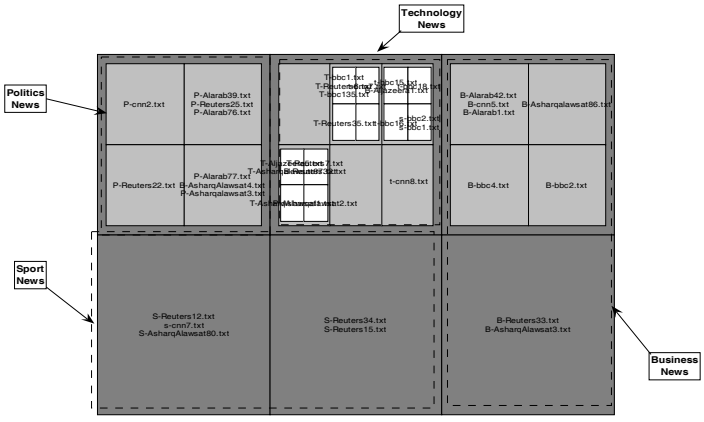


Fig. 11. The resulting 3-Layer GHSOM map.

5. Conclusion

Based on the experiments performed in this research, it can be concluded that the quantization error is a map quality measure which is data-dependent. It measures the map in terms of the given data. Typically, the quality of the map is measured in terms of the training data. From the experiments, lesser quantization error was obtained by GHSOM than by SOM which means that the data sent to the GHSOM is well located in the map. When training SOM and GHSOM, the training time taken by using the SOM algorithm was longer compared to the training time achieved by the GHSOM, as the factor of vector size increases. This indicates that the SOM algorithm is not useful for large number of training set documents. As the SOM algorithm cannot handle the hierarchies of the data, it is much difficult for the user to keep an overview of the various clusters. Also, the data hierarchies could not be discovered by the algorithm. We have evaluated the classification quality of SOM and GHSOM using Precision, Recall and F1 measures based on the results obtained by the algorithms. The results have shown that the GHSOM performed better than the SOM in Precision and Recall results for the four main clusters that were used in the training data. Also, when calculating the F1 measures, the GHSOM obtained better results than the SOM. The inclusion of word sense disambiguation on Arabic script documents that will be used to identify the semantic of documents is subject for future research.

6. References

Aalarab (2007). Aalarab news. www.alarab.com. 2007.

Adafre, Rijke (2006). Finding Similar Sentences across Multiple Languages in Wikipedia. 11th Conference of the European Chapter of the Association for Computational Linguistics, pp. 62-69, Trento, Italy, April 2006, Association for Computational Linguistics, Morristown, NJ, USA.

Aljazeera (2007). Aljazeera news. www.aljazeera.net. 2007.

Asharqalawsat (2007). Asharqalawsat news. www.asharqalawsat. 2007.

BBC (2007). BBC news. www.bbc.com. 2007.

Chen, R. Chau, C. Yeh (2004). Discovering Parallel Text from the World Wide Web. Proceedings of the second workshop on Australasian information security, Data Mining and Web Intelligence, and Software Internationalisation, pp 157–161, Dunedin, New Zealand, 2004, Australian Computer Society, Inc. Darlinghurst, Australia.

CNN (2007). CNN news. www.cnn.com. 2007.

Dittenbach, Rauber, D. Merkl(2001). Business, Culture, Politics, and Sports - How to Find Your Way Through a Bulk of News? On Content-Based Hierarchical Structuring and Organization of Large Document Archives. Proceedings of the 12th P International Conference on Database and Expert Systems Applications, pp 200 – 210, 3-540-42527-6, Munich, Germany, September 2001,Springer-Verlag London, UK.

Evans (2005). Identifying Similarity in Text: Multi-Lingual Analysis for Summarization. Ph.D. Thesis. Columbia University. 2005.

Helmut, Dieter (2005). A comparison of support vector machines and self-organizing maps for e-mail categorization. Proceedings of the 4th Australasian Data Mining Conference (AusDM'05), pp 189-204, 1-86365-716-9, Sydney, Australia, December 2005, University of Technology Sydney, Australia.

Jin, Barrire (2005). Exploring sentence variations with bilingual corpora. Corpus Linguistics 2005 conference. Birmingham, United Kingdom, July 2005, NRC Institute for Information Technology, Canada.

Kohonen (1990). The self-organizing map. Proceedings of the IEEE, pp 1464 – 1480, 0018-9219, September 1990, IEEE.

Lee, Yang (2003). A Multilingual Text Mining Approach Based on Self-Organizing Maps. Journal of applied intelligent. Volume 18, No.3, (May 2003), page numbers (295-310), 0924-669X.

Liu, Wang, Zheng (2005). Mental tasks classification and their EEG structures analysis by using the growing hierarchical self-organizing map. Neural Interface and Control, 2005, Proceedings of the First International Conference, 115- 118, 0-7803-8902-6, May 2005, IEEE.

Michael Binyon(2008). *September 11th and the Western Media and Cross - Cultural Misunderstanding Role of Dialogue between Arab And West Seminar*, Kuwait, 2008.

Selamat A. and Omatu S. (2004). Feature Selection and Categorization of Web Pages Using Neural Networks, Int. Journal of Information Sciences, Elsevier Science Inc. Vol. 158, (January 2004), page number (69-88).

Selamat A., Choon N-C, Abu Bakar A.Z., Mikami Y.(2007), Arabic Script Web Documents Language Identification Using Decision Tree-ARTMAP Model, 2007 International Conference on Convergence Information Technology (ICCIT 2007), pp. 21-23, November 2007, Gyeongju-si, Gyeongbuk, Korea.

Tangsripairoj, Samadzadeh (2005). Organizing and visualizing software Repositories using the growing hierarchical Self-organizing map. Proceedings of the 2005 ACM symposium on Applied computing SAC, pp. 1539- 545, 1-58113-964-0, Santa Fe, New Mexico, 2005, ACM, New York, NY, USA.

Xafopoulos A., Kotropoulos C., Almpanidis G. and Pitas I(2004). Language Identification in Web Documents Using Discrete HMMs. Pattern Recognition. Vol. 137, No. 3, March2004, page numbers(583-394), 0031-3203.

Zamir (1999). Clustering Web Documents:A Phrase-Based Method for Grouping Search Engine Results. University of Washington: Ph.D.Thesis.

Zhai, Shah (2005).Tracking News Stories Across Different Sources.Proceedings of the 13th annual ACM international conference on Multimedia, pp. 2-10, 1-59593-044-2, Hilton Singapore, 2005, ACM, New York, NY, USA.

22

Self-organizing Maps in Web Mining and Semantic Web

Emil Şt. Chifu and Ioan Alfred Leţia
Technical University of Cluj-Napoca
Romania

1. Introduction

The nature inspired approaches represent a new trend in computer science in general and in the Semantic Web, due to their scalability and robustness. Neural networks represent one category of nature inspired solutions. The self-organizing map (SOM) is a very popular unsupervised neural network model (Kohonen, et al., 2000). It is a data mining and visualization method for complex high dimensional data sets.

In the first part of the chapter, we present how the SOM model can be applied in Web mining, by giving sets of documents as input data space for SOM. The result of applying SOM on a set of documents is a map of documents, which is organized in a meaningful manner so that documents with similar content appear at nearby locations on the two-dimensional map display. From the information retrieval point of view, our implemented SOM-based system creates document maps that are readily organized for browsing. A document map also clusters the data, resulting in an approximate model of the data distribution in the high dimensional document space. Some experimental results are included, where a couple of meaningful clusters have been discovered by our system in a subset of the "20 newsgroups" data set (Lang, K., 1995). The clustering capability of our system allows users to find out quickly what is new in a Web site of interest by comparing the clusters obtained from the site at different moments in time.

In the rest of the chapter, we focus on how a more complex SOM based unsupervised neural network model is used for enriching a domain ontology. Building complete and reliable domain ontologies is the basis for the success of the Semantic Web. The ontology enrichment process consists in the addition of new concepts which will be attached as hyponyms for the existent nodes of the ontology (Pekar and Staab, 2002). The names of the new concepts are terms represented linguistically by common noun phrases. The enrichment process can also add new instances to existent concepts of the ontology. In this case, the process is also known in the literature as ontology population or named entity classification, where the named entities are represented linguistically by proper names of people, organizations, locations etc. (Cimiano and Völker, 2005). In both cases, the process is algorithmically the same, the only difference being the grammatical category of the linguistic entities to be classified: common noun phrases representing terms for new concepts to be added or proper noun phrases representing named entities, i.e. new instances for the existent

concepts. The noun phrases representing terms and named entities are extracted from a domain text corpus by a text mining process. For every noun phrase, the mining acquires a vector that encodes contextual content information, in a distributional vector space. The enrichment behaves like a classification of the terms or named entities into the taxonomy of the given ontology, based on a similarity metric in the distributional vector space.
The growing hierarchical self-organizing map (GHSOM) model consists of a set of SOM maps arranged as nodes in a hierarchy and it is able to discover hierarchical clusters (Dittenbach et al., 2002). The SOM's in the nodes can grow horizontally during the training by inserting either one more row or one more column of neurons. The SOM's in the nodes can also grow vertically during the training, by giving rise to successor nodes. Like the classical Kohonen SOM model, GHSOM is an unsupervised neural network. Unsupervised hierarchical neural models in general start the growing of a dynamic tree-like topology from a single initial node. We propose a new neural network model, called Enrich-GHSOM, as an extension of the GHSOM system, which allows the growing to start from an initial tree. The growth of the hierarchy proceeds along with the predefined paths of the given hierarchy. Consequently, our model allows a classification of the data items into an existing taxonomic structure that plays the role of an initial state for the tree-like neural network model. In order to apply our model in ontology enrichment, the taxonomy that is subject to enrichment is given as the initial state of the hierarchical self-organizing map. So, an essentially symbolic knowledge structure – taxonomic tree – is converted into a neural representation as an initial state of the hierarchical self-organizing map. The actual taxonomy enrichment takes place via an unsupervised training of the neural network by exposing the initialized hierarchical self-organizing map to the vector representation of the terms extracted from the domain corpus. A reverse, neural-symbolic translation is done after this enrichment process. This is the knowledge extraction step, and the output is the final enriched taxonomy. Our taxonomy enrichment framework is a hybrid one, as it has to deal with neural-symbolic integration. The neural-symbolic translations in both directions have been naturally achieved, since our framework merely operates upon the taxonomic structure of the ontology, which is in agreement with the hierarchical structure of the self-organizing neural network.
The ontology enrichment experiments that will be presented are in the "Lonely Planet" tourism domain. The taxonomies, the corpus, and the named entities are the ones proposed in the PASCAL ontology learning and population challenge (Grobelnik et al., 2006). The evaluation of the enrichment is based on cross-validation and on gold standard ontologies. Our experimental results prove that the quality of the ontology enrichment is considerably improved by using our semantics based vector representations for the classified (newly added) terms, like the document category histograms and the document frequency times inverse term frequency (DF-ITF) weighting scheme.

2. Self-organizing Maps

The self-organizing maps have been created by Teuvo Kohonen as a particular kind of neural networks (Kohonen, et al., 2000). There are multiple views on SOM; the different definitions are the following. SOM is a model of specific aspects of biological neural nets (the ordered "maps" in the cortex). SOM is a model of unsupervised machine learning and an adaptive knowledge representation scheme. SOM is a tool for statistical analysis and

visualization: it is both a projection method which maps a high dimensional data space into a lower dimensional one and a clustering method so that similar data samples – represented as vectors of numerical attribute values – tend to be mapped on nearby neurons. The resulting lower dimensional output space is a two-dimensional grid of arrays (the SOM map) which visualizes important relationships among the data, – which are latent in the input data set – in an easily understandable way. This dimensionality reduction maintains the topology of the input vectors, i.e. inputs that are close to each other – in other words, similar – in the input space are also close to each other in one of the clusters of the map.
In short, SOM is a data mining and visualization method for complex high dimensional data sets. Even though there are no explicit clusters in the input data set, important relationships are nevertheless latent in the data. SOM can discover and illustrate these latent structures of an arbitrary data set. SOM can describe different aspects of a phenomenon in any domain, provided that the data in the domain can be represented by vectors of numerical attributes.
The map learns by a self-organization process. No a priori knowledge about the membership of any input data item (vector) in a particular class or about the number of such classes is available. Hence, the training proceeds with unlabeled input data like any unsupervised learning. The clusters (classes) are instead discovered and described with gradually detected characteristics during the training process.
The map consists of a regular two-dimensional (rectangular) grid of processing units – the neurons. Each unit has an associated model of some multidimensional observation, represented as a vector of attribute values in a domain. SOM learning is an unsupervised regression process which consumes at every iteration one available observation represented as a vector of values for the attributes in a given domain. The role of a learned map is to represent all the available observations with optimal accuracy by using a restricted set of model vectors associated to the map units.

2.1 The Learning Algorithm

The initial values for the model vectors – also referred to as reference vectors or weight vectors – of the map units can either be chosen depending on the problem domain or they can be taken randomly. Each iteration of the learning algorithm processes one input (training) vector (one sample) $\mathbf{x}(t)$ as follows. Like usually for unsupervised neural networks, some form of a competitive learning takes place: the winner unit index c, which best matches the current input vector, is identified as the unit where the model vector is the most similar to the current input vector in some metric, e.g. Euclidean:

$$\| \mathbf{x}(t) - \mathbf{m}_c(t) \| \leq \| \mathbf{x}(t) - \mathbf{m}_i(t) \| , \tag{1}$$

for any unit index i. Then all the model vectors or a subset of them that correspond to units centered around the winner unit c – i.e. units in the neighbourhood area of c –, including the winner itself, are adjusted in the direction of the input vector, as follows:

$$\mathbf{m}_i(t+1) = \mathbf{m}_i(t) + \mathbf{h}_{ci}(t) * [\mathbf{x}(t) - \mathbf{m}_i(t)] , \tag{2}$$

where h_{ci} is the neighbourhood function, which is a decreasing function on the distance between the i-th and c-th units on the map grid, and whose maximum value corresponds to

$i = c$. In practice, the neighbourhood area is chosen to be wide in the beginning of the learning process, and both its width and height decrease during learning. The updating in (2) forms a globally ordered map in the process of learning.
A map unit has six immediate neighbours in a hexagonal map topology, which is usually the preferred topology. This is only a hexagonal lattice type of the two-dimensional array (grid) of neurons, so the SOM map continues to be a planar rectangle. The hexagonal neighbourhood topology effect is gained by shifting correspondingly rows number 1, 3, 5, … of the rectangular map to the right and keeping rows 0, 2, 4, … untouched. A rectangular topology corresponds to a rectangular lattice, and a map unit has only four immediate neighbours. Consequently, the number of neighbour units affected during the learning is only four as compared to six in the hexagonal topology.
After the training of a map, its reference vectors have converged to stationary values and the result is a topology-preserved map. Similar reference vectors become close to each other, and dissimilar ones become far from each other on the map. Moreover, two input data items which are close to each other in the input data space are mapped onto the same or neighbouring neurons on the map. Each neuron together with its own reference vector represents similar data items of the input space, and a set of neighbouring neurons with similar reference vectors creates a cluster.

2.2 Cluster Visualization

A subset of data items which are close to each other in a high dimensional input data space – and thus defines a cluster in the input space – are arranged to a map area consisting of neurons close to each other also in the two-dimensional SOM display. As a consequence, the problem of discovering a cluster in a high dimensional data set with the help of the self-organizing maps reduces to the problem of discovering the map area whose neurons to contain all the data in the cluster. Actually, we have to find the boundaries of the map cluster. Finding the boundaries of a SOM map cluster is based on applying the unified-distance matrix (U-matrix) algorithm on a SOM map (Ultsch, 1993).
U-matrix visualizes the map in grey-levels, in order to express how similar or dissimilar adjacent neurons are (Wilppu, 1997; Hautaniemi, et. al., 2003). In a hexagonal self-organizing map topology, six hexagons (extra neurons) around each neuron separate geometrically the neuron from its six immediate neighbours and show its similarity with each of them. The lighter a separating hexagon, the bigger the similarity of the reference vectors of the two separated neurons, and the darker the hexagon, the bigger the dissimilarity of the reference vectors. This way, SOM map clusters can be discovered visually as “valleys” or “depressions” (light areas) separated by “hills” (dark areas or borders). Moreover, the higher (i.e. darker) a hill separating two clusters, the more dissimilar the clusters in the multidimensional input data space.
In (Ultsch, 1993), an older (in fact the original) version of the U-matrix algorithm is used, by calculating at each map unit the sum of the distances of the reference vector of that neuron to the reference vectors of the immediate neighbouring neurons.

3. Web Mining with Self-organizing Maps

Applying SOM on natural language data means doing data mining on text data, for instance Web documents (Lagus, 2000). The role of SOM is to cluster numerical vectors given at input and to produce a topologically ordered result. The main problem of SOM as applied to natural language is the need to handle essentially symbolic input such as words. If we want SOM to have words as input then SOM will arrange the words into word categories. But what about the input (training) vector associated to each input word? What should be the vector components, i.e. the attributes of a word? Similarity in word appearance is not related to the word meaning, e.g. "window", "glass", "widow".

We have chosen to classify words by SOM, creating thus word category maps. The attributes of the words in our experiments were the count of the word occurrences in each document in a collection of documents. Consequently, we have chosen to represent the meaning of each word as related to the meanings of text passages (documents) containing the word and, symmetrically, the semantic content of a document as a bag-of-words style function of the meanings of the words in the document. The lexical-semantic explanation of this contextual usage meaning of words is that the set of all the word contexts in which a given word does and does not occur provides a set of mutual constraints that captures the similarity of meaning of words and passages (i.e. documents, contexts) to each other. The measures of word-word, word-passage and passage-passage relations are well correlated with several cognitive phenomena involving semantic similarity and association (Landauer, et. al., 1998). The meaning of semantically similar words is expressed by similar vectors.

After training a SOM on all the words in a collection of documents – where the vectorial coding of words represents the contextual usage –, the result self-organizing map groups the words in semantic categories. There are also other possibilities to code words, which lead to grammatical or semantic word categories (Honkela, 1997; Kohonen, et.al., 1996; Kohonen, et. al., 2000).

3.1 System Architecture

The architecture of our system is based on two self-organizing maps. The first one creates a semantically ordered spread of all the word forms in a large collection of Web documents. This is also called the map of word categories or level 1 SOM. The second SOM (called document map or level 2 SOM) represents a semantically ordered spread of all the documents in the collection, where the documents are codified as vectors that are histograms of word categories. The word categories are the ones as already induced into the word category map units. For every word category, the histogram representation of a document contains the number of word form occurrences in the document which belong to that word category. This way we have reduced the dimensionality of the document vectors from thousands of components which would correspond to thousands of different word forms in a classical bag-of-words approach. The dimensionality is reduced to around 200 or 300 components which correspond to 200 or 300 different word categories, enough to express the number of different concepts in a shallower or wider domain. Thus the reduced dimensionality removes the noise caused by the variability in word usage; since the number of dimensions is much smaller than the number of word forms, minor differences in terminology will be ignored. Our category based approach is able to solve the terminology problem in information retrieval, i.e. the problem of possibly different terminologies used in

the documents and in a user query for one and the same concept, in other words, the problem of synonymy or near-synonymy.
The aim of our system is to classify the document collection by using the criterion of semantic similarity. Hence the graphical browsing interface of our system is in essence a document map (level 2 SOM).

3.2 System Implementation

The system is written in C and bash script. We have used the LEX software package (Lesk, and Schmidt, 1975) for implementing the preprocessing module, which reads and counts the word occurrences in all the documents in a collection, by ignoring all the HTML tags. The preprocessing module also ignores 450 common words, i.e. English words having no semantic load. These words have been taken from the information retrieval software package GTP (Giles, et. al., 2003). Finally, the preprocessing also means a stemming phase that uses a morphological analyzer for English, which is part of the GATE system (Cunningham, et. al., 2002). The stemming is done in order to reduce the number of word forms by keeping only their stem.
The SOM_PAK (Kohonen, et. al., 1996) system is used for the training of all our SOM maps. The result of training the document SOM is a text file containing for every document category a list of document names that belong to that category, i.e. the list of documents managed into the corresponding map unit. The format of this text file is exemplified with seven document categories in Fig. 1, where each row corresponds to a different map unit. The first two integer numbers in each row represent the rectangular coordinates (x and y) of the current unit. The document category name follows the coordinates of the unit and becomes the identification label of the unit. The document category name is given by the name of the first document in the (training) data set that "hit" the unit during the training process. This name occurs as the last in the enumeration of document names in the category, after the colon.
All the seven document categories in Fig. 1 are semantically related as they all contain as documents emails from one and the same newsgroup (*talk.politics.mideast*) in the "20 newsgroups data set" (Lang, 1995). The seven corresponding map units are neighbours on the document map and they constitute together an area or *cluster*. The aggregation of the neurons in this cluster is noticeable from their coordinates and from the hexagonal topology adopted.

3.2.1 Graphical User Interface

The graphical interface has been implemented by using the PHP language. The system creates the graphical interface as an interactive graphical display that is implemented as a dynamically created HTML file. This PHP module reads the text file of document categories (created by the document SOM and exemplified in Fig. 1) and translates this document classification into a dynamical HTML file which is the graphical display of the document map itself. Every map unit is labelled with the associated document category name, which is found out as explained at the beginning of the current section (Section 3.2). A better alternative would be to label a map unit with the most relevant and representative document in the corresponding category, i.e. the name of the document whose vector is closest to the model vector of that unit (Lagus, et.al., 1999). A second label on each map unit

represents the number of documents in the corresponding category. For instance, the map unit for the last document category in Fig. 1, having coordinates 9, 5 on the map, is also labelled 6.

```
 8  3 75381  :  75381
 8  4 75382  :  75369  75382
 9  4 75394  :  75371  75389  75394
10  4 75393  :  75393
11  4 75400  :  75370  75392  75400
 8  5 75395  :  75395
 9  5 75399  :  176854  75366  75387  75388  75390  75399
```

Fig. 1. Example document categories.

The interface allows the navigation on the document map from any Web browser. The aim of this browsing is the retrieval of relevant documents in two steps. Click on a map unit gives access to the index of documents in that unit, which is also a dynamically generated HTML file, containing a list of links, each of which having as text the document name and pointing to the document itself. Then click on a document name in this list allows viewing that document.

3.3 Experimental Evaluation

The experiments reported here take as test data the "20 newsgroups data set" (Lang, 1995). This data set contains 20,000 UseNet news postings having the form of email messages. The 20,000 messages were collected at random from 20 different Netnews newsgroups, 1000 messages from each newsgroup (Lang, 1995). The data set is "labelled", by being already partitioned into twenty categories. This labelling helped us in evaluating the clustering results of the same set of email documents as discovered and visualized by our document SOM. In one of our most successful experiments, we have selected randomly 40 documents from each newsgroup, summing up a total set of 800 message documents. This balanced subset of the original "20 newsgroups" data set has been taken as input data space for our SOM-based system in order to arrive at an email document SOM map.

An important question in this experiment was to choose a size for the SOM map, in order to arrive at a map with the highest degree of visual expressiveness for clustering (Wilppu, 1997). The map size means the total number of neurons of the rectangular grid. For a given data set, different map sizes mean different granularity levels, in terms of the average number of data items to belong to a neuron. If the map is too small, it is too rough and consequently it might hide some important differences that should be detected in order to separate the clusters. This is because too many unsimilar data items could belong to the same neuron. When the map is too big, then it is too detailed and, besides the important differences, the map displays also too small differences, which are often unimportant for

clustering. This is because data items which are very similar could belong to different neurons, when normally we expect them to belong to one single neuron.
We have chosen a map size of 16 (columns) times 12 (rows) considered as suitable for the input data space of 800 data items (800 email documents). This also conforms to the suggestions in (Wilppu, 1997), where a suite of experiments with input data sets of different cardinality and different SOM map sizes is described. Fig. 2 shows the result email document SOM map image, where grey levels occur as an effect of applying the U-matrix algorithm for cluster visualization. The U-matrix algorithm used here is included in the SOM_PAK program package (Kohonen, et. al., 1996), which is part of our system. The algorithm conforms to the description in Section 2.2.

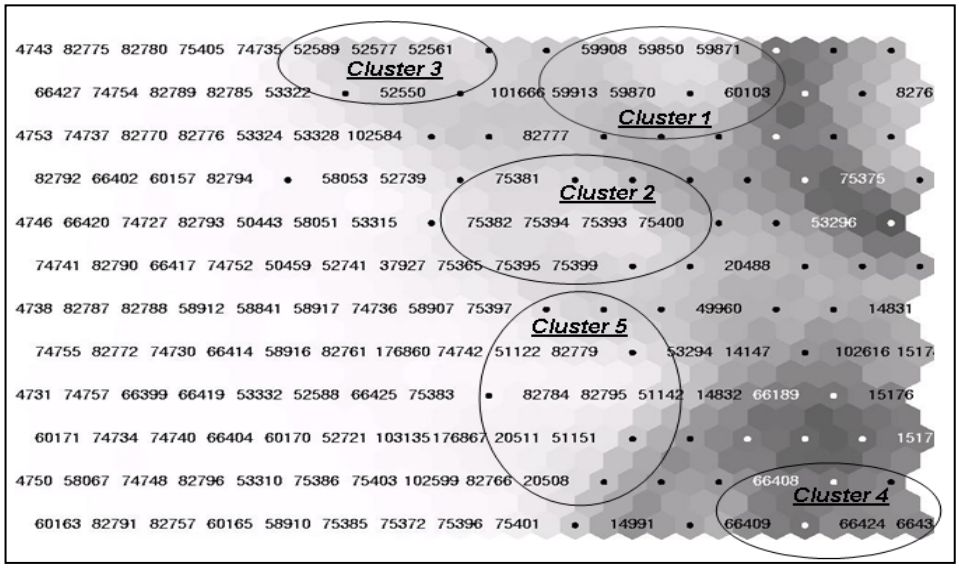


Fig. 2. Document SOM map for 800 email messages taken from the "20 newsgroups" data set

3.3.1 Clustering Results

The document map in Fig. 2 clearly illustrates four clusters discovered by the map. Neurons in Cluster 1 contain 15 email messages, which all belong to the newsgroup *science.space*. Only 15 out of a total of 40 messages in the *science.space* newsgroup in the input space were discovered to belong to Cluster 1. Hence, even if the *accuracy* of this cluster is 100%, its coverage is only 15/40, so 37.5%.
All the 17 documents in Cluster 2 belong to the newsgroup *talk.politics.mideast*, but there are 41 messages in the input data that belong to this newsgroup. Cluster 2 contains seven neurons whose explicit description in terms of email messages grouped as document categories in each neuron is given in Fig. 1. Actually only 40 input messages are "officially" labelled by the authors of the "20 newsgroups" data set to belong to the newsgroup *talk.politics.mideast*. One more message, named 176854, and found out by our map to belong to Cluster 2, has been "abusively" put by the authors into another newsgroup, namely

talk.politics.misc. The header of this email indicates explicitly `Newsgroups: talk.politics.mid-east, misc.headlines, talk.politics.misc`.
Similarly, Cluster 3 contains 12 messages, 11 of them from the newsgroup *rec.sport.hockey*. This cluster is less clearly bordered on the map, because of the semantic overlap with other messages some of them form the related newsgroup *rec.sport.baseball*. In fact, the only message in Cluster 3, which is outside of the expected newsgroup *rec.sport.hockey*, is from the related newsgroup *rec.sport.baseball*. Finally, Cluster 4 on the map represents 11 messages, 10 of them from the newsgroup *comp.windows.x*, and one from the related newsgroup *comp.sys.ibm.pc.hardware*. Table 1 shows the classification quality parameters accuracy and coverage associated with the four clusters.

Cluster No.	**Newsgroup**	**Accuracy** (Correct/Predicted)	**Coverage** (Correct/Actual)
1	*Science.space*	100% (15 / 15)	37.5% (15 / 40)
2	*talk.politics.mideast*	100% (17 / 17)	41.5% (17 / 41)
3	*rec.sport.hockey*	91.5% (11 / 12)	27.5% (11 / 40)
4	*comp.windows.x*	90.9% (10 / 11)	25% (10 / 40)
5	Combination of *talk.religion.misc*, *soc.religion.christian*, *alt.atheism*	80.8% (21 / 26)	17.5% (21 / 120) where 120 = 3*40

Table 1. Classification accuracy and coverage associated with document clusters on Fig. 2

3.3.2 Discussion of Results

There are some more results found out from our document map induced from 800 news messages, and illustrated in Fig. 2. For instance, there is one more cluster, Cluster 5, also mentioned in Table 1, which contains 26 email messages, 21 of them being a mixture of messages from three different newsgroups: *talk.religion.misc*, *soc.religion.christian*, and *alt.atheism*. The first two newsgroups are obviously related to each other, and they are also semantically related with the third, even if this relation sounds more like an antonymy. Similar topics are nevertheless discussed in messages about religion and atheism.
About 85% of the 800 email messages are contained in about the left half of the map, which is completely white, and constitutes a huge cluster. Such a cluster has no clear semantic content, because it contains messages from all the 20 newsgroups, including the messages left out from the five clusters already mentioned. The technical explanation for this phenomenon is that the document SOM map was unable to display semantic differences in this big cluster. The differences in the semantic content of the messages could be too small when the authors of the messages use too few words specific to the domain of the newsgroup or sometimes when they communicate announcements with no bearing with the domain of the newsgroup.
Another explanation for the huge cluster is that the majority of the email messages in the "20 newsgroups" data set are addressed to many different real newsgroups. The more newsgroups a message is addressed to, the more arbitrary its inclusion (by the authors of the "20 newsgroups" data set) in one of the 20 groups, and the fewer semantic differences discernable by our SOM-based system for such messages.
Our clustering results were worse when we didn't ignore the 450 stop words mentioned in Section 3.2, because these words with no semantic load introduced noise that reduced the

capability of displaying semantic differences between email message documents. We have also examined some word category SOM maps. One of our first interesting results with word maps is that, when we didn't ignore the stop words, the word category maps contained isolated (unclustered) neurons representing stop words, which were situated only near the margins of the map. The explanation is that the word map separates the stop words from the other content-rich words, the latter being contained in the interior of the map.

4. Growing Hierarchical Self-organizing Maps

GHSOM is an extension of the Self-organizing Map (SOM, also known as Kohonen map) learning architecture (Kohonen, et al., 2000). Data spaces contain some latent structuring in the form of clusters. SOM maps can discover and illustrate this clustering. However, some *hierarchical structures* are also latent in data sets. To give an interesting example in the present context, a thesaurus is a data space consisting of terms in a language, represented as a lexical database. The main relation between the terms in a thesaurus is the taxonomic relation. However, because of their essentially flat topology, SOM maps have a limited capability to discover and illustrate hierarchical clusters in data sets. A solution for this problem is represented by the *hierarchical SOM maps*.

The growing hierarchical self-organizing map model consists of a set of SOM maps arranged as nodes in a hierarchy and it is able to discover hierarchical clusters (Dittenbach, et. al., 2002). The SOM's in the nodes can grow horizontally during the training by inserting either one more row or one more column of neurons. This happens iteratively until the average data deviation (quantization error) over the neurons in the SOM map decreases under a specified threshold τ_1. For one neuron, the quantization error is the dissimilarity of all the vectors of the data items mapped into the neuron versus the weight vector of the neuron.

The SOM's in the nodes can also grow vertically during the training, by giving rise to successor nodes. Each neuron in the SOM map could be a candidate for expansion into a successor node SOM map (see Fig. 3). The expansion takes place whenever the data deviation on the current neuron is over a threshold τ_2. This sounds like a zoom into the data subspace mapped into the parent neuron, because the successor SOM map is trained merely with data items in that subspace. Further node expansions continue recursively on successor nodes, and the training of the whole GHSOM model finally stops (converges) when both thresholds are satisfied. The training begins with a single-neuron SOM map having the whole input data set mapped into its only neuron. This becomes the root of the final, completely trained GHSOM model.

The thresholds τ_1 and τ_2 control the granularity of the hierarchy learned by GHSOM in terms of depth and branching factor. A low τ_1 with a much lower τ_2 leads to a deep hierarchy with an increased number of neurons into the SOM nodes, and consequently an increased branching factor also. A high τ_1 with a lower τ_2 leads to deep hierarchies with small SOM nodes (with few neurons), and consequently a reduced branching factor corresponding to the reduced number of neurons in SOM nodes. When both thresholds are low and comparable, then the hierarchy is flat with a high branching factor. If both thresholds are high and comparable, then the hierarchy is flat with a low branching factor.

Each level in a learned GHSOM model displays a more detailed clustering of the data space as compared to the parent level. This corresponds to a top-down process of hierarchical clustering of the input data space items.

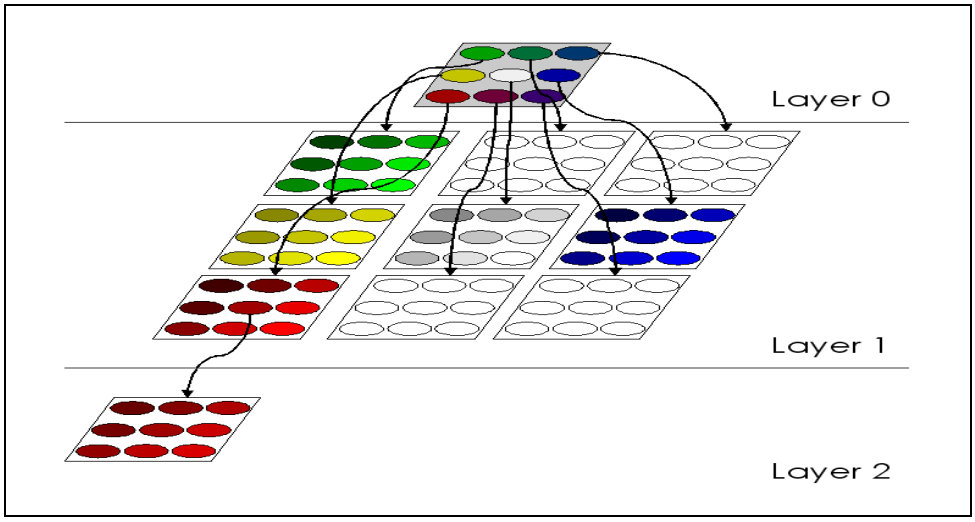


Fig. 3. The GHSOM neural network model.

5. Enrich-GHSOM

The growth of a GHSOM is a completely unsupervised process, being only driven by the unlabeled input data items themselves together with the two thresholds and some additional learning parameters. There is no way to suggest from outside any initial paths for the final learnt hierarchy. We have extended the GHSOM model with the possibility to force the growth of the hierarchy along with some predefined paths of a given hierarchy. Our new extended model, *Enrich*-GHSOM, is doing a classification of the data items into an existing tree hierarchy structure. This initial tree plays the role of an initial state for the tree-like neural network model. The classical GHSOM model grows during the training by only starting from a single node. The top-down growth in our extended model starts from a given initial tree structure and inserts new nodes attached as successors to any of its intermediate and leaf nodes.

In Enrich-GHSOM, the nodes of the predefined hierarchy are labelled with some data item labels from the input data space used for training. The training data items propagate top-down throughout the given tree hierarchy structure. When the propagation process hits a parent SOM of a tree node, then the weight vector of the corresponding parent neuron in that parent SOM is initialized with the data item vector of that successor node label. The weight vectors of the SOM neurons with no successor are initialized with random values. Then the training of that SOM proceeds by classifying the training data items against the initialized neurons. Training data items that are similar (distributionally similar as vectors) to the predefined initialized neurons are propagated downwards to the associated successor SOM nodes to continue the training (recursively) on that predefined successor SOM. Data

items that are not similar to the initialized neurons are mapped to other, non-initialized, neurons in the same SOM, and they are not propagated downwards into the predefined hierarchy. They remain as mapped into that SOM, and are considered as classified into the parent neuron of that SOM, i.e. as successor of that parent.
For instance, consider the parent neuron of a current SOM node is labelled *mammal*, and there are two predefined successor nodes labelled *feline* and *bear*, which correspond to two predefined initialized neurons in the current SOM. Then the training data item vector *dog* is not similar to any of the two neuron initializer weight vectors associated to *feline* and *bear* (see Fig. 4, where the neuron initializers are marked with bold).

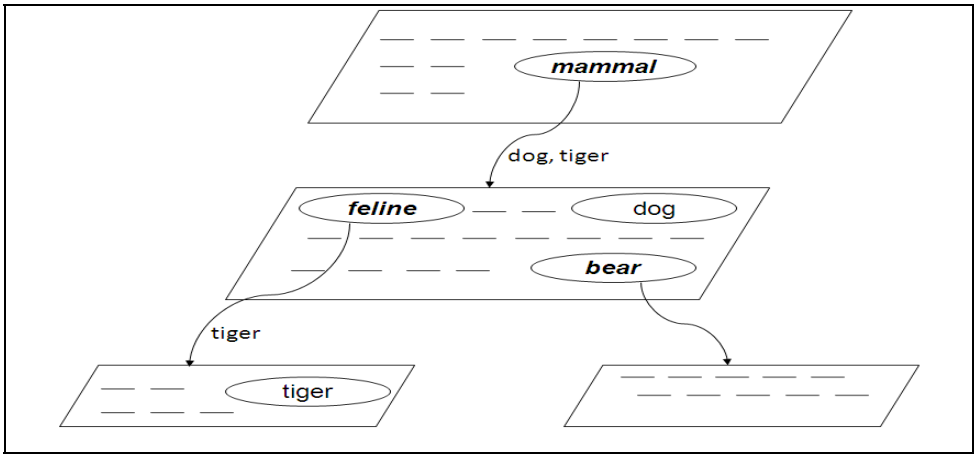


Fig. 4. The Enrich-GHSOM neural network model.

So *dog* will remain as classified into that SOM – mapped on another, non-initialized neuron – i.e. as successor (hyponym) of *mammal* and twin of the existent nodes *feline* and *bear*. Also, a data item labelled *tiger* – similar with the weight vector of the predefined "*feline*" neuron – will be propagated into the associated predefined successor SOM map together with other terms that correspond to *felines*, which will all become direct or indirect hyponyms of the concept *feline*. The process continues top-down for all the SOM nodes in the predefined initial tree hierarchy, ending at the leaves. The data item vector representations of the labels of the given initial tree play the role of *predefined initializer weight vectors* of our neural model.

6. Text-Based Ontology Enrichment Using Hierarchical Self-organizing Maps

The most important prerequisite for the success of the Semantic Web research is the construction of complete and reliable domain ontologies. Building ontologies is still a time consuming and complex task, requiring a high degree of human supervision and being still a bottleneck in the development of the semantic web technology.
The process of domain ontology enrichment has two inputs, an existing ontology – which plays the role of background knowledge – and a domain text corpus. The aim of our work is to automatically adapt the given ontology according to a domain specific corpus. We enrich

the hierarchical backbone of the existing ontology, i.e. its taxonomy, with new domain-specific concepts extracted from the corpus (Pekar and Staab, 2002).
Our framework for taxonomy enrichment is based on an extended model of hierarchical self-organizing maps, which represent an unsupervised neural network architecture. The candidates for labels of newly inserted concepts are terms collected by mining a text corpus. The term extraction process is based on recognizing linguistic patterns (noun phrases) in the domain corpus documents. Each term encodes contextual content information, in a distributional vector space. The context features of a term are the frequencies of its occurrence in different documents of the corpus. The classification of the extracted terms into the taxonomy of the given ontology proceeds by associating every term to one target node of the taxonomy, based on a similarity in the distributional vector space. That term becomes a new concept added to the taxonomy, and it is attached as hyponym (successor) under the target node.
Unsupervised hierarchical neural models in general start the growing of a dynamic tree-like topology from a single initial node. Our neural network model, called *Enrich-GHSOM*, is an extension of one of these existent systems, GHSOM (Dittenbach, et al., 2002), and it allows the growing to start from an initial tree. This is suitable to the knowledge structure to be enriched – a taxonomy, i.e. an *is-a* hierarchy of concepts. The taxonomy that is subject to enrichment is given as the initial state of the hierarchical self-organizing map. So, an essentially symbolic knowledge structure – taxonomic tree – is converted into a neural representation as an initial state of the hierarchical self-organizing map. The actual taxonomy enrichment takes place via an unsupervised training of the neural network by exposing the initialized hierarchical self-organizing map to the vector representation of the terms extracted from the domain corpus. A reverse, neural-symbolic translation is done after this enrichment process. This is actually the knowledge extraction step whose output is the final enriched taxonomy. Our taxonomy enrichment framework is a hybrid one, as it has to deal with neural-symbolic integration. The neural-symbolic translations in both directions have been naturally achieved, since our framework merely operates upon the taxonomic structure of the ontology, which is in agreement with the hierarchical structure of the self-organizing neural network.

6.1 Related Work on Taxonomy Enrichment

There are two main categories of approaches for taxonomy enrichment (Buitelaar, et al., 2005b): methods based on *distributional similarity* and *classification of terms into an existing taxonomy* on one hand, and approaches using *lexico-syntactic patterns*, also known as Hearst patterns (Hearst, 1992), on the other hand. Our enrichment approach belongs to the former category.
In the term classification approach, the terms extracted from a domain specific corpus of text are classified into an existent taxonomy (Pekar and Staab, 2002; Cimiano and Völker, 2005; Alfonseca and Manandhar, 2002; Witschel, 2005; Widdows, 2003). In a top-down variant of this classification (Pekar and Staab, 2002; Alfonseca and Manandhar, 2002; Witschel, 2005), there is a top-down search on the existent taxonomy in order to find a node under which a new term is to be inserted as a successor (hyponym). The classification of the terms is made according to a similarity measure in a distributional vector space. Each term is represented as a vector with information about different contexts of its occurrences in the corpus.

The top-down classification behaviour in our framework is modelled by a growing hierarchical self-organizing map (GHSOM) architecture (Dittenbach,et al., 2002) extended with the possibility to set an initial state for the tree-like neural network. In our new extended neural model, called Enrich-GHSOM, the given taxonomy is set as the initial state of the neural network. The model allows to classify the extracted terms into the existing taxonomy by attaching them as hyponyms for the intermediate and leaf nodes of the taxonomy. Details of this process are given in section 7.2.
A similar, although non top-down approach is (Widdows, 2003). There is a search for a node to attach a new concept as a hyponym of, by finding a place in the existent taxonomy where the corpus derived semantic neighbours of the candidate concept are most concentrated. He supposes that at least some of the semantic neighbours are already in the taxonomy, and he defines a function to compute the class label for the set of neighbours – a hypernym for all the neighbours. This class label becomes the concept under which to attach the new term as hyponym. The similarity measure to find neighbours is based on a latent semantic analysis vector space (Landauer and Dumais, 1997).

7 A Neural Model for Unsupervised Taxonomy Enrichment

The architecture of our framework for taxonomy enrichment is implemented as a pipeline with several linguistic and machine learning processing stages. The whole processing can be divided in two main steps: the *term extraction* step and the *taxonomy enrichment* step.

7.1 Extraction of Terms

The candidates for the labels of new concepts inserted during the taxonomy enrichment are terms representing *noun phrases*, identified by mining the domain text corpus. In order to identify the terms by a linguistic analysis of the corpus documents, our framework relies on several processing resources offered by the ANNIE module for analyzing English texts in the GATE framework (Cunningham, et al., 2002): morphological analyzer (stemmer), tokenizer, sentence splitter, the Hepple part-of-speech tagger, and a JAPE (Cunningham, et al., 2002) transducer. The transducer has the role to identify noun phrase constructs, based on regular expressions over different parts of speech of the component words.

7.2 Taxonomy Enrichment

The terms extracted from the domain text corpus are mapped to classes (concepts) of the existing taxonomy. The taxonomy enrichment algorithm proceeds by "populating" the given taxonomy with the terms collected from the corpus. The *Enrich*-GHSOM neural network drives a top-down hierarchical classification of the terms along with the given taxonomy branches and inserts new nodes (concepts) corresponding to these classified terms. Every new concept is attached as successor of an intermediate or a leaf node of the given taxonomy and becomes a hyponym of that target node.
In order to use our Enrich-GHSOM neural network to induce such a taxonomy enrichment behaviour, a symbolic-neural translation is first done by parsing a textual representation of the initial taxonomy (*is_a*(*concept*, *superconcept*) assertions or OWL format). The result of this parsing is the initial internal tree-like state of the neural network. In order for the initialized network to be able to classify terms into this initial taxonomic structure, apart from the

vector representation of the classified terms, a representation as a numerical vector is also needed for each node in the initial taxonomy. This vector plays the role of initial weight vector for the neural network (see section 5). It is the vector representation for the noun phrase concept label associated to the node, computed as will be described in section 7.3. The acquisition of this vector takes place in the same way as the acquisition of the vector representation of the classified terms (section 7.3).
We assume that the concept labels of the initial taxonomy are terms – noun phrases – extractable from the domain text corpus from which the classified terms themselves have also been extracted. Their vectors are then computed in the same way as the vectors of all the corpus extracted terms which are classified during the enrichment. Using the same corpus from a specialized domain to acquire the feature vectors of the concepts in the initial taxonomy and the terms to be classified is a reasonable choice, since it will reduce the problems with ambiguous (multiple) senses of one and the same term.

7.3 Vector Representation for Terms

Since Enrich-GHSOM is a connectionist system, the terms classified by Enrich-GHSOM and the concepts of the given taxonomy have to be represented as vectors. In our framework, the attributes (features) of the vector representation of a term or concept encode contextual content information, in a *distributional vector space*. Specifically, the context features are the frequencies of the occurrence of the term – classified term or concept label term – in different documents of the corpus. The number of component attributes of such a term vector coincides with the number of documents of the text corpus out of which all the terms have been extracted. Every attribute in the vector of a term is essentially the number of occurrences of the term in one document. This representation is inspired from the latent semantic analysis (Landauer and Dumais, 1997). A similar semantics-based dimensionality reduction effect as the one obtained in the latent semantic analysis by singular value decomposition is achieved in our framework by the *document category histograms* (DCH), defined in what follows.
The vector representation in the current framework satisfies Harris' distributional hypothesis (Cimiano and Völker, 2005; Buitelaar, et al., 2005b): the meaning of each classified term (or concept label) is related to the meanings of the contexts in which the term (or the concept label) occurs. In such a setting, we use the *distributional similarity* which asserts that the meaning of semantically similar terms and concept labels is expressed by similar vectors in the distributional vector space. The Euclidean distance is used in the current framework to compute the dissimilarity among vectors.
The framework allows multiple ways to encode the frequencies of occurrence, starting from simple *flat counts of occurrences*. Another variant is the *DF-ITF weighting scheme*, which means "document frequency times inverse term frequency". We propose this weighting scheme, which is a transposed of TF-IDF (Buitelaar, et al., 2005a) relative to a term/document occurrence matrix. TF-IDF is used in document classification (text categorization) and information retrieval. Now we rather classify terms, by using DF-ITF. By using this weighting scheme, we consider that long documents, which talk about too many terms, should have a lower weight when classifying terms, since they have a reduced discrimination power among the meanings of different terms. This effect is achieved by our DF-ITF weighting scheme and is confirmed by the experimental results reported in section 7.4.

A third way to encode the vector representation is one in which we propose the vector to be a *document category histogram* (*DCH*). Specifically, first a SOM (Kohonen, et al., 2000) is trained having the corpus documents as input data space to arrive at approximately 200 semantic document categories. Documents similar in meaning are clustered together by the unsupervised SOM neural network. In this SOM training, the documents are represented as vectors of frequencies for the terms they talk about. Equally like the term vectors, the document vectors are collected from the same term/document matrix, but after transposing this matrix. As we want a number of approximately 200 semantic document categories, we impose the training of a rectangular SOM map of dimension 16x12. Then, by summing up the frequencies of a term in different documents of the same category, and merely keeping the summed frequencies in different document categories as vector components, we arrive at a reduced dimensionality for the vector representation. In our experiments reported in section 7.4 with the "Lonely Planet" tourism data set, the reduction induced by such a vector representation as a histogram on semantic document categories is from 1801 (which represents the number of documents in the "Lonely Planet" corpus) to a value between 175 and 180 (in the different experimental runs described in section 7.4).

7.3.1 Data Sparseness

The *dimensionality reduction* achieved by using the document category histogram (DCH) representation is important since it removes the semantic noise caused by minor differences in semantic content for different corpus documents. Such documents now belong together to the same semantic category. This intuition is already confirmed by our experiments reported in previous work (Chifu and Leția, 2006). Moreover, the term/document occurrence matrix is sparse (with many zeros), and reducing the dimensionality by using histograms leads to less sparse vectors. A more natural behavior of the neural network model is expected by using reduced and less sparse vectors.

A source of data sparseness is represented by *terms with very few occurrences* in the text corpus. Among such terms are the most generic terms that label the roots of the main trees in a given initial taxonomy and usually the concepts which are very high in a taxonomy. When in the *Enrich-GHSOM* neural network such an overly generic term with a very sparse vector labels the concept of one of the roots, and also when using the flat count vector representation instead of the histogram representation, then the main tree rooted by that concept is unable to attract and classify a relevant quantity of training terms. Thus the top-down search during the classification is misled. It is the case of the root concepts *spatial_concept*, *intangible*, and *thing* in the ontology of the "Lonely Planet" tourism dataset used in the present experiments. Some of the branches of these main trees are populated by no training term, which leads to the *starvation* of the neural network. Starvation means that the neural network enters an infinite loop when trying to tune the quantization error on a neuron below the thresholds (see section 4). Many of our experiments which used a flat count vector representation failed by starvation. As opposed, all the experiments using the reduced, histogram vector representation (DCH) converged to a result.

A way of reducing the number of zeros in the vector representation of the generic terms that label the generic concepts in the initial taxonomy is the *centroid vector* (Pekar and Staab, 2002; Cimiano and Völker, 2005). We have used the idea of centroid of a concept in the following way: the average vector of the vector representations of all the concepts in the sub-tree rooted by the given concept, including the root itself. Using the centroid representation

method has led us to a significant improvement of the experimental results, partially reported in (Chifu and Leţia, 2006), where we rather proposed a similar approach: one of the more specific concepts in a main tree becomes a substitute for the too generic concept in the root of the tree. So, the label of every main tree root was one representative and more specific concept in the tree, for instance *course* was a substitute for *activity*, and *staff* was a substitute for *person* (in the "4 universities" domain). The improvement obtained in related work by using the centroid vector representation for concepts is reported in (Pekar and Staab, 2002; Cimiano and Völker, 2005).

7.4 Experimental Results

The experiments carried out in what follows are in the tourism domain, consisting of a corpus and a given taxonomy (the "Lonely Planet dataset") (Grobelnik, et al., 2006). The associated corpus consists of 1801 text descriptions of tourist destinations from different countries around the world.

7.4.1 Experimental Setup

In order for the corpus extracted terms to actually become domain specific concepts, they have to be noun phrases with enough frequency of occurrence in the domain specific corpus. In the term extraction process, we have set a threshold for the extracted noun phrases to occur in at least 0.5% of the number of documents in the corpus. Having set this frequency threshold, we have extracted and acquired the corresponding numerical vector representations for 1241 noun phrases. These extracted terms are classified against the taxonomy of a tourism ontology consisting of 72 concepts, which is proposed in the PASCAL ontology learning and population challenge (Grobelnik, et al., 2006).

The evaluation of the enrichment means evaluating the quality of the mapping from corpus extracted terms into target concepts of the given initial taxonomy. An extracted term becomes a new concept added to the taxonomy, and it is attached as hyponym (successor) under its associated target node. In order to evaluate the taxonomy enrichment, we followed a *cross-validation strategy* (Pekar and Staab, 2002; Witschel, 2005; Widdows, 2003). In every experimental run, exactly one node in the given initial taxonomy of 72 concepts was removed from the taxonomy, together with the whole subtree rooted by that node. The classification process was run against the result taxonomy, and the position of the held out concept, as classified like any corpus extracted term is assessed. The correct (direct hit) classification of the concept corresponds to its initial position in the taxonomy before its removal. In other words, the concept should be mapped to a target concept which was its direct hypernym (parent node) before its experimental removal. The process should be repeated 71 times, for every concept in the taxonomy except its very root, named root. Actually we repeated this experimental run 43 times, since we only had corpus statistical data to build the distributional vector representation for 43 of the taxonomy concepts. (We need a statistical distributional vector for every term to be classified.)

7.4.2 Evaluation Measures

The most appropriate measure for evaluating the taxonomy enrichment task is the *learning accuracy*, defined and evaluated in (Grobelnik, et al., 2006; Cimiano and Völker, 2005; Pekar & Staab, 2002, Alfonseca and Manandhar, 2002; Witschel, 2005). By choosing this measure,

we consider correct classifications of the new concepts with different levels of detail. For instance, the new concept *cat* can be mapped to the target concept *feline*, *carnivore*, *mammal*, or *animal* with different levels of detail, as a consequence of different taxonomic distances between the target concept as chosen by the system and the direct hypernym of the classified concept before its removal. Before removal, *cat* was direct hyponym of the *feline* concept. Classifying *cat* as *feline*, by associating it to the *feline* target concept is a direct hit, since *cat* is correctly a *direct hyponym* of *feline*, i.e. 100% learning accuracy. Though, classifying *cat* as *carnivore*, *mammal*, or *animal* are near hits, since *cat* is correct only as an indirect hyponym of *carnivore*, *mammal*, or *animal*, corresponding say to 50%, 30%, 20% learning accuracy respectively.
For a given classified term *i*, if *pi* is the target concept assigned (predicted) by the system, and *ci* the ccorrect target concept, the learning accuracy is the average over all the classified terms i of the function $LA(pi, ci)$, where the function LA is defined as

$$LA(p, c) = \frac{\delta(top, a) + 1}{\delta(top, a) + \delta(a, c) + \delta(a, p) + 1} \tag{3}$$

top is the root of the taxonomy, and *a* is the least common subsumer of the concepts *p* and *c* (i.e. the most specific common hypernym of *p* and *c*). $\delta(x,y)$ is the taxonomic distance between the concepts *x* and *y*, i.e. the number of taxonomy edges to be traversed when going from the taxonomy node labelled *x* towards node *y*. This is the most used formula to compute the learning accuracy. In the context of the Pascal ontology learning and population challenge, it is actually called *symmetric learning accuracy*, and the term *learning accuracy* is used for a historically initial version of the learning accuracy measure, as introduced by (Hahn and Schnattinger, 1998):

$$LA'(p, c) = \begin{cases} \dfrac{\delta(top, a) + 1}{\delta(top, c) + 1} & \text{if } p \text{ is ancestor of } c \\ & \text{(then also } a = p\text{)} \\ \dfrac{\delta(top, a) + 1}{\delta(top, a) + 2 * \delta(a, p) + 1} & \text{otherwise} \end{cases} \tag{4}$$

According to formulae (3) and (4) to compute both variants of the learning accuracy, the same number of edges in the taxonomic distance between the predicted and the correct target concept means a better accuracy when the edges are lower in the taxonomy. This is due to the intuition that the same number of edges between two concrete (lower in the taxonomy) concepts means an increased similarity (a reduced semantic distance), as compared to the same number of edges between two abstract concepts (higher in the taxonomy).
Another quantitative evaluation measure similar in spirit to the learning accuracy is the *edge measure*. It actually counts the average deviation (in terms of taxonomic distance) between the system predicted target concept and the correct one. Consequently, as opposed to the

first two learning accuracy measures (formulae (3) and (4)), the edge measure means a better classification for a lower edge measure value.

7.4.3 Evaluation Results

A first set of experimental runs is based on a document category histogram (DCH) vector representation for the extracted terms and concept label terms. Also, the concept label terms of the given initial taxonomy are represented using the *centroid* method for the whole sub-tree of a given concept node, as described in section 7.3. The improvements gained by using DCH and centroid are already confirmed qualitatively by our experiments reported in (Chifu and Leţia, 2006). Furthermore, not only the training of the Enrich-GHSOM neural network is less efficient on flat count vectors with 1801 attributes (corresponding to the 1801 corpus documents) compared to the 180 attributes (for the 180 semantic document categories) in DCH's, but also using flat count (unreduced) vectors often leads to the starvation of the neural network.

In a second set of experiments, we first applied the *DF-ITF* weighting scheme on the flat count term vectors of 1801 attributes. The result vectors were then converted into DCH histograms, thus reducing the term vector dimensionality to 179.

(Cimiano and Völker, 2005) and (Pekar and Staab, 2002) used the centroid vector to reperesent the concept nodes. (Pekar and Staab,2002) found out that their best results were achieved when taking into account only the first three levels of successors in the sub-tree of the concept in order to compute the centroid. The experiments in (Cimiano and Völker, 2005) considered only the direct successors of the concept to compute the centroid. Driven by these results, we ran a third set of experiments, in which we considered only the first level of successors to represent the centroid of any concept in the given taxonomy, like in (Cimiano and Völker, 2005). We didn't also try the three-level version of (Pekar and Staab, 2002), since the results would be similar with our results for whole sub-trees. This is because the average depth of the taxonomy to be enriched in our experiments is 4, and the majority of the nodes don't have sub-trees of depth greater than 3. In this third set of experiments we kept the DF-ITF and DCH settings like in the second experiment.

All these experiments involved the cross-validation experimental strategy described in section 7.4.1, in which every classified concept was previously removed together with its whole subtree. In a fourth set of experiments, we only removed the tested concept alone, and kept untouched its whole subtree, which rather became a subtree of its parent node.

We evaluated the three learning accuracy measures on placing the 43 concepts in their actual position in the given initial ontology from the Pascal challenge (Grobelnik, et al., 2006). The results are illustrated in Table 2.

All the three learning accuracy measures are considerably improved by using the DF-ITF weighting measure, and keeping the DCH histogram vector representation. These results prove that the quality of the enrichment is improved by using our contributed semantics based vector representations (DCH and DF-ITF) for the classified terms and the concept label terms in the initial taxonomy. Another finding is that limiting the depth of the sub-concepts for the computation of the centroid vector representation for taxonomy concepts leads to a slight degradation of the learning accuracy. The experiments in (Witschel, 2005) also confirm that using whole sub-trees to represent the centroid of the concepts improves the performance of the taxonomy enrichment. Removing only the tested taxonomic node alone keeps the enrichment quality roughly unchanged as compared to removing the whole

subtree. This confirms the expectation that a concept is semantically linked with all the nodes in its subtree. The whole subtree represents a class of terms which lexicalize the concept in the root of the subtree.

Vector Representation	DCH	DF-ITF+DCH	DF-ITF+DCH	DF-ITF+DCH
Concept Label Centroid	whole subtree centroid	whole subtree centroid	first-level centroid	whole subtree centroid
Removed	whole subtree	whole subtree	whole subtree	node only
Learning Accuracy	33.351%	39.654%	37.679%	38.76%
Symmetric Learning Accuracy	33.998%	40.272%	38.016%	40.402%
Edge Measure	3.023	2.651	2.861	2.698

Table 2. Learning accuracy of the taxonomy enrichment when using DCH, DF-ITF, and different variants of centroid.

7.4.3.1 Named Entity Classification

In a last set of experiments, instead of classifying terms represented by common noun phrases extracted from the "Lonely Planet" corpus, we rather classified noun phrases for proper names – i.e. named entities – extracted from the same corpus. The majority of the named entities occur few times in the corpus, and many of them only occur once, in a singe document. This is why, in the experiments reported in what follows, we have reduced the frequency threshold to zero. It was 0.5% in the preceding experiments (see section 7.4.1).

Having no more frequency threshold for the corpus extracted noun phrases, we found and extracted a total of 43006 noun phrases, compared to 1241 in the preceding three taxonomy enrichment experiments. Some of them are common nouns and the other are named entities. We will refer in what follows to this experiment as *the maximal experiment*. To reduce the dimensionality of the data, and consequently the inherent noise, one of our experiments was trying to keep only what is absolutely necessary for the classification. We kept a minimum of common noun phrases corresponding to the concept labels in the taxonomy, and a minimum of proper noun phrases representing the set of named entities asked to be classified in the PASCAL ontology learning and population challenge (Grobelnik, et al., 2006). The total number of common and proper noun phrases extracted is reduced to 631. We will call this experimental run *the minimal experiment*.

We evaluated these last experiments automatically by using the PASCAL challenge site online evaluation system. This evaluation system is based on a *gold standard*, i.e. an ontology populated with the set of named entities that are asked to be classified in the PASCAL challenge. In other words, the PASCAL competition target set of named entities are considered as correctly mapped to the different concepts in the gold standard ontology. In *the maximal experiment*, a number of 625 named entities extracted from the "Lonely Planet" corpus are classified against an ontology consisting of 72 concepts, which is proposed in the PASCAL challenge (Grobelnik, et al., 2006). Actually there are much more named entities extracted by our framework, but only 625 of them are also included in the set of named

entities asked to be classified in the PASCAL ontology learning and population challenge. In *the minimal experiment*, 417 named entities are classified into a taxonomy consisting of 96 concepts. Table 3 illustrates these last two experiments, as evaluated automatically with the PASCAL challenge online evaluation system.

There are two explanations for the lower classification quality values in the maximal experiment as compared to the minimal one. First, the minimal experiment uses the DCH histogram vector representation as compared to the flat counts of the maximal experiment, and second is the noise caused by the much bigger quantity of noun phrases classified in the maximal experiment – 43006 versus 631. Also, an explanation for an overall degraded quality of the *named entity classification* as compared to the *taxonomy enrichment* in the preceding experiments is that the classified named entities have very low frequency of occurrence as compared to the classified terms (common nouns) from the taxonomy enrichment, and consequently they have a very sparse vector representation. This misleads their classification.

Experiment	*maximal experiment*	*minimal experiment*
Vector Representation	flat counts	DCH
Concept Label Centroid	whole subtree centroid	whole subtree centroid
Learning Accuracy	22.4%	31.2%
Symmetric Learning Accuracy	21.2%	28.5%
Edge Measure	3.754	4.767

Table 3. Learning accuracy of the named entity classification.

8. Taxonomy Learning Using Self-Organizing Trees

SOTA (self-organizing tree algorithm) (Herrero, 2001) is another dynamical tree-like self-organizing map. It is an unsupervised neural network with a binary tree topology, which is available as SOTArray (Herrero, 2001). The clustering algorithm in SOTA is a top-down process: the tree grows starting from its root, and then develops into more detailed classifications on the lower hierarchical levels. This growing stops when a predefined level of classification detail is reached. The level of detail is set according to the distribution of probability obtained by randomization of the data set to be classified. The tree-like output space can freely grow until adapting as much as possible to the variability of the input data space. Alternatively, new nodes can grow until reaching a complete classification of the data items, i.e. until having a single data item in every leaf of the tree. This is the setting we used when applying SOTA for taxonomy learning.

We have used the SOTA model to learn a taxonomy starting from a text corpus. A learned SOTA hierarchy plays the role of a learned taxonomy. The hierarchical clustering of the terms is done in a top-down manner, the upper levels being generated before the lower levels, which are more detailed and will contain more specific terms. SOTArray classifies the initial data set only in the leaves of the binary tree that it develops, the inner nodes being empty. The taxonomy structure obtained with the SOTA algorithm is a binary tree of terms. In every leaf we have one term from the corpus. After the training of SOTArray, we labelled the inner nodes starting from the leaves and ascending towards the root of the tree

hierarchy, by searching the WordNet database (Fellbaum, 1998) for the most specific common hypernym of every two sibling nodes.

9. Conclusions and Further Work

This chapter presented several uses of the self-organizing maps in the context of web mining and the semantic web. The self-organizing maps constitute a powerful model for Web mining by defining a visual overview of a set of Web documents. A document SOM map is a semantically ordered spread of the documents in the set. Our SOM-based visualization system for document collections is a powerful information retrieval tool for browsing a set of Web documents. The system is especially useful when the user has rather limited knowledge about the domain or the contents of the text collection.

The unsupervised top-down neural network based approach and framework for taxonomy enrichment is also essentially based on self-organizing maps. The framework can be applied to different domains and languages. The experimental results obtained in the "Lonely Planet" tourism domain prove that our contributed semantics based vector representations, i.e. the *document category histograms* and the *DF-ITF weighting scheme*, are suitable for the task of taxonomy enrichment.

The comparison of taxonomy enrichment systems (and of named entity classifiers) is problematic. Different systems use different domains and, even for the same domain, they use different corpora of different sizes and different ontologies. (Grobelnik, et al., 2006) present such a comparison of existent systems, and the conclusion is that the classification quality degrades with the increase in the size of the ontology.

Another interesting point is that sometimes given taxonomic structures are not reflecting correctly some fine-grained meanings. For instance, in the initial taxonomy used in our experiments, f*orest* is hyponym of *area*. However the context in which the term *forest* occurs in the corpus are rather specific to plants (*plant* concept), which is far in the taxonomy from *area*. Our system "incorrectly" classified *forest* as *plant*.

The data sparseness remains a problem for the task of taxonomy enrichment. Terms (or named entities) represented by sparse vectors have an increased chance to be wrongly classified, because of their reduced power of being attracted towards the correct branches and nodes of the taxonomy. Thus the top-down search during the classification is misled, and this phenomenon is mostly encountered in the case of named entity classification, where named entities have very sparse vector representations. Consequently, as further work, we will try to change the statistical distributional vector representation of the terms to further reduce the dimensionality of the vectors. We will try using pseudo-syntactic dependencies as representation of the terms, in the spirit of (Cimiano and Völker, 2005).

10. References

Alfonseca, E., Manandhar, S. (2002). Extending a lexical ontology by a combination of distributional semantics signatures, In: A. Gómez-Pérez, V.R. Benjamins (eds.) *13th International Conference on Knowledge Engineering and Knowledge Management*, LNAI. Springer, pp. 1-7

Buitelaar, P., Cimiano, P., Grobelnik, M., Sintek, M. (2005a). Ontology learning from text, Tutorial at ECML/PKDD workshop on Knowledge Discovery and Ontologies

Buitelaar, P., Cimiano, P., Magnini B. (2005b). Ontology learning from text: an overview, In: P. Buitelaar, P. Cimiano, B. Magnini (eds.) *Ontology Learning from Text: Methods, Evaluation and Applications, Frontiers in Artificial Intelligence and Applications Series,* IOS Press, pp. 1-10

Chifu, E.Şt., Leţia, I.A. (2006). Unsupervised ontology enrichment with hierarchical self-organizing maps. In: Leţia, I.A. (ed.) *IEEE 2nd International Conference on Intelligent Computer Communication and Processing*, pp. 3-9

Cimiano, P., Völker, J. (2005). Towards large-scale, open-domain and ontology-based named entity classification, In: *RANLP'05, International Conference on Recent Advances in Natural Language Processing*, pp. 166-172

Cunningham, H., Maynard, D., Bontcheva, K., Tablan, V. (2002). GATE: a framework and graphical development environment for robust NLP tools and applications, In: *40th Anniversary Meeting of the ACL*

Dittenbach, M., Merkl, D., Rauber, A. (2002). Organizing and exploring high-dimensional data with the Growing Hierarchical Self-Organizing Map, In Wang, L., et al. (eds.) *1st International Conference on Fuzzy Systems and Knowledge Discovery*, vol. 2, pp. 626-630

Fellbaum, Chr. (Ed.) (1998). *WordNet: An Electronic Lexical Database*, MIT Press. Cambridge, Mass.

Giles, J.T.,L. Wo, L., and Berry, M.W. (2003). GTP (General Text Parser) software for text mining, in H. Bozdogan, ed., *Statistical Data Mining and Knowledge Discovery*, CRC Press, Boca Raton, pp. 455-471.

Grobelnik, M., Cimiano, P., Gaussier, E., Buitelaar, P., Novak, B., Brank, J., Sintek, M. (2006). Task description for PASCAL challenge, Evaluating ontology learning and population from text

Hearst, M.A. (1992). Automatic Acquisition of Hyponyms from Large Text Corpora, *In: 14th International Conference on Computational Linguistics*, pp. 539-545

Hahn, U., Schnattinger, K. (1998). Towards text knowledge engineering, In: *15th National Conference on Artificial Intelligence and the 10th Conference on Innovative Applications of Artificial Intelligence* (*AAAI/IAAI*), pp. 524-531

Hautaniemi, S., Yli-Harja, O., Astola, J., Kauraniemi, P., Kallioniemi, A., Wolf, M., Ruiz, J., Mousses, S., and Kallioniemi, O.-P. (2003). Analysis and visualization of gene expression microarray data in human cancer using self-organizing maps, *Machine Learning*, vol. 52, pp. 45-66.

Herrero, J., Valencia, A., and Dopazo, J. (2001). A hierarchical unsupervised growing neural network for clustering gene expression patterns. *Bioinformatics*, 17, pp. 126–136.

Honkela, T. (1997). Self-organizing maps in natural language processing, *PhD thesis*, Neural Networks Research Center, Helsinki University of Technology, Finland.

Honkela, T., Kaski, S., Lagus, K., and Kohonen, T. (1996). Exploration of full-text databases with self-organizing maps, in *Proceedings of the International Conference on Neural Networks*, vol. I, pp. 56-61.

Kohonen, T., Hynninen, J., Kangas, J., and Laaksonen, J. (1996). SOM_PAK: The self-organizing map program package, *Technical Report A31*, Helsinki University of Technology, Laboratory of Computer and Information Science, 1996.

Kohonen, T., Kaski, S., Lagus, K., Salojärvi, J., Honkela, J., Paatero, V., Saarela, A. (2000). Self-organization of a massive document collection, *IEEE Transactions on Neural Networks* 11, pp. 574-585

Lang, K. (1995). NewsWeeder: Learning to filter news. In: *12th International Conference on Machine Learning*, pp. 331-339

Lagus, K., (2000). Text retrieval using self-organized document maps, *Technical Report A61*, Helsinki University of Technology, Laboratory of Computer and Information Science.

Lagus, K. and Kaski, S. (1999). Keyword selecton method for characterizing text document maps, *in Proceedings of the 9th International Conference on Artificial Neural Networks*, vol. 1, pp. 371-376.

Landauer, T.K., Foltz, P.W. and Laham, D (1998). Introduction to Latent Semantic Analysis, Discourse Processes, vol. 25, 1998, pp. 259-284.

Landauer, T., Dumais, S., (1997). A solution to Plato's problem: the latent semantic analysis theory of acquisition, induction and representation of knowledge, *Psychological Review* 104, 211–240

Lesk M.E. and Schmidt, E.,(1995). Lex – a lexical analyzer generator, *Computing Science Technical Report 39*, AT&T Bell Laboratories, Murray Hill, 1975; UNIX Programmer's Manual, vol. 2B, Bell Laboratories.

Pekar, V., Staab, S. (2002). Taxonomy learning – factoring the structure of a taxonomy into a semantic classification decision, In: *COLING'02, 19th International Conference on Computational Linguistics*, pp.786-792

Ultsch, A., (1993). Self organized feature maps for monitoring and knowledge acquisition of a chemical process", in S. Gielen and B. Kappen, eds., *Proceedings of the International Conference on Artificial Neural Networks,* pp. 864-867.

Widdows, D. (2003). Unsupervised methods for developing taxonomies by combining syntactic and statistical information, In: *HLT-NAACL Conference*, pp. 197-204

Wilppu, E. (1997). The visualization capability of self-organizing maps to detect deviations in distribution control, *Technical Report* 153, Turku Centre for Computer Science.

Witschel, H.F. (2005). Using decision trees and text mining techniques for extending taxonomies, In: *Learning and Extending Lexical Ontologies by using Machine Learning Methods*, Workshop at ICML-05, pp. 61-68

Secure Wireless Mesh Network based on Human Immune System and Self-Organizing Map

Mahira M. Mowjoon and Johnson I Agbinya
Centre for Real time Information Networks,
Faculty of Engineering and Information Technology,
University of Technology, Sydney
Australia

1. Introduction

Wireless Mesh Networks (WMNs) are evolving as a future generation wireless mobile networks and bring the dream of connected world into reality. According to 802.11s standard Nodes in a mesh network can be divided into four classes, Client or Station (STA), Mesh Point (MP), Mesh Access Point (MAP) and Mesh Portal Point (MPP). Client is a node that requests services but does not contribute in path discovery, Mesh Point (MP) is a node that participates in the mesh operations, Mesh Access Point (MAP) is a MP attached to an access point (AP) to provide services for clients (STA), and Mesh Portal Point (MPP) is a MP with additional functionality to act as a gateway between the mesh and an external network (Hidenori et al., 2006). Fig 1 shows WMN architecture according to 802.11s standard.

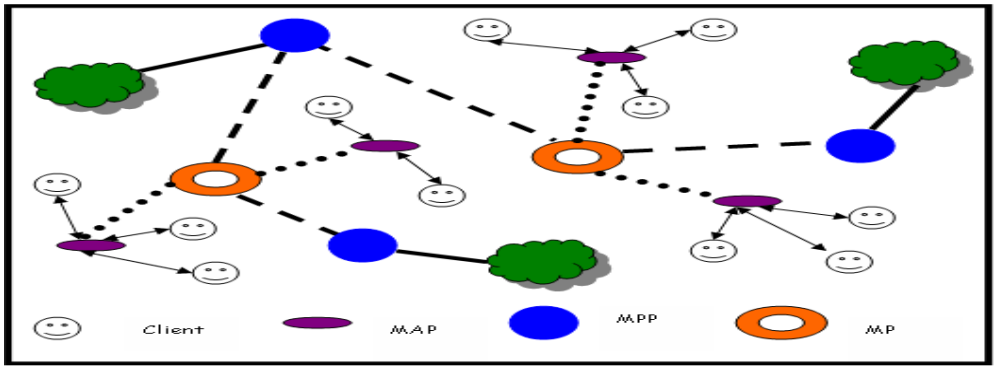


Fig. 1. WMN Architecture

Further, WMN is dynamically self-organized and self configured, with the nodes in the network automatically establishing and maintaining mesh connectivity among them. Major

principles of self-organization can be classified in to four categories specifically local state evaluation, interaction between individuals, negative feedback loop and positive feedback loop. This concept is common to both biological systems and communication systems and the figures 2 and 3 compare the concept of feedback loops for biological system and the feedback loop for WMN system respectively. Positive feedback loop amplifies an effect whereas negative feedback loop controls the system behaviour. Any dependencies and global control are prevented by acting upon local information. Moreover, direct and indirect information exchange is used to update local state and interact with the system environment. The self organizing nature of WMN brings many rewards such as low up-front cost, easy network maintenance, robustness, and reliable service coverage and delivers wireless services for extended applications namely real time intelligent transportation systems, spontaneous networks, rural networks, community and neighbourhood networks, broadband home networks, building automation, security surveillance systems, metropolitan area networks and health and medical systems.

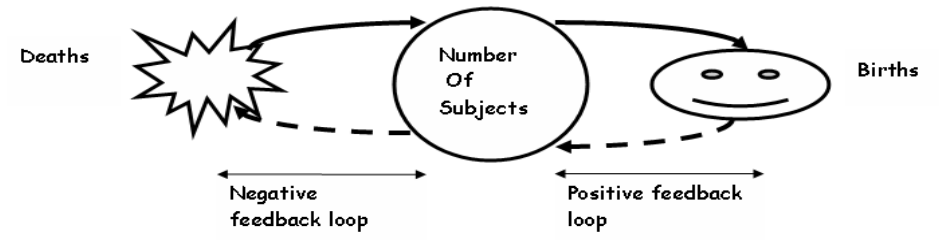


Fig. 2. Feedback loops in biological system

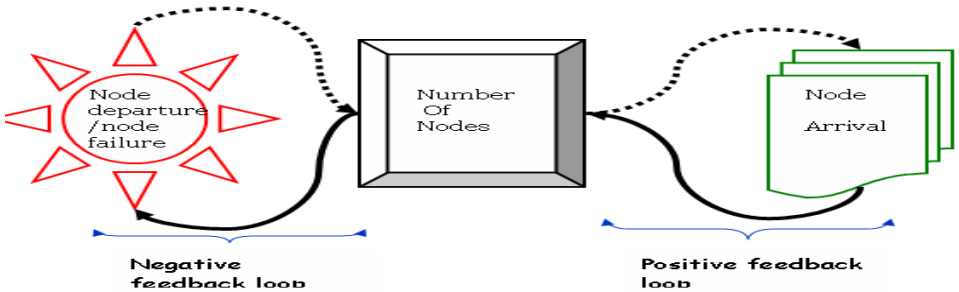


Fig. 3. Feedback loops in WMN system

Although there are some similarities between Mobile Ad-Hoc Networks and Wireless Mesh Networks there are also number of important differences. In contrast with Mobile Ad-Hoc Networks, static nodes, essentially Mesh Access Points in Mesh networks communicate with each other over wireless links.

1.1 Application scenarios of Wireless Mesh Networks

Real time intelligent transportation system

Mesh networking technology can be extensively used to provide useful passenger information services in buses, ferries, and trains and can support remote monitoring of in-vehicle security and driver communications. Fig. 4. shows real time intelligent transportation system.

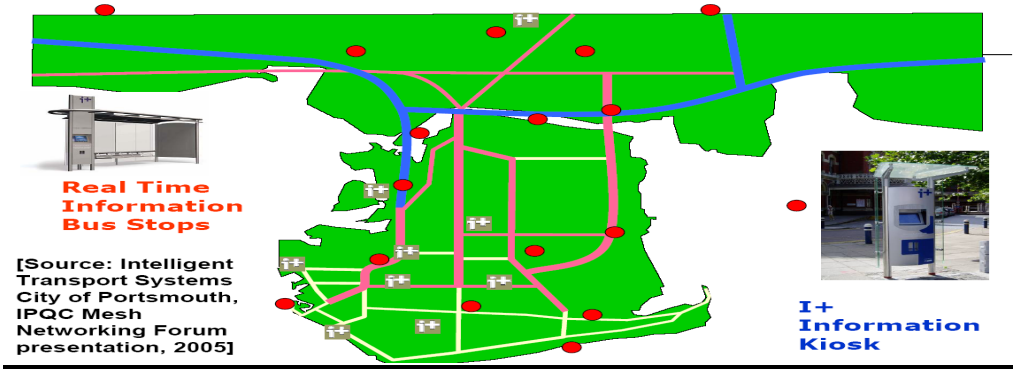


Fig. 4. Real time intelligent transportation system

Spontaneous (emergency/disaster) network

WMNs facilitate group of people to establish group networks during an emergency situation, where the people involved have no knowledge about the environment. This service is offered by simply placing wireless mesh access points in desired locations. Fig. 5. illustrates emergency network.

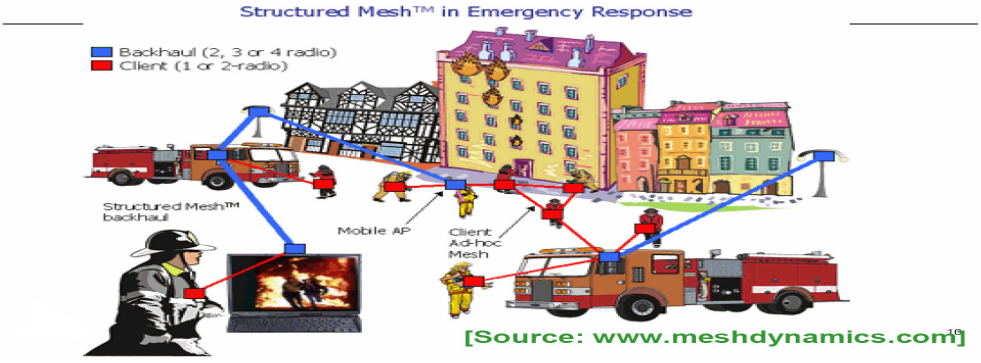


Fig. 5. Emergency network

Rural network

Since the communication between nodes does not depend on the wired backbone, WMN is a cost effective option in under developed regions and covers vast area than the community networks.

Community and neighborhood network
WMNs resolve problem of resource utilization in community networks and covers large areas between houses while reducing the network access cost and provide multiple paths to access Internet and offer flexible connectivity with neighbours.
Broadband home network
In home networking, WMN offers location independent services and provides flexible and robust connectivity of the network.
Building automation
In large buildings controlling and monitoring of various electrical devices are very common. Currently, these tasks are executed through expensive wired networks, by deploying WMN controlling and monitoring can be performed at lower cost.
Security surveillance system
Security is becoming vital in critical environments and security surveillance systems become a necessity for enterprise buildings, shopping malls, grocery stores, etc. In order to setup such systems at locations as required, WMNs are a much more feasible solution than wired networks to connect all devices.
Health and medical system
Critical and bulk data processing and transmission are the key issues in medical centres and hospitals. While traditional wired networks provide only certain services wireless mesh technology offers efficient transmission of high resolution medical images and large volume of monitoring information.

2. Wireless Mesh Network security

Despite of hot move in WMN research, substantial investigation is needed to address the most challenging factors such as security, inter operability, Quality of service, connectivity, scalability and compatibility. Even though security is the major factor that affects the deployment of WMN it is often a secondary reflection in development, thus considerable research is still needed to address the security aspect. Recently, there has been a rapid boost in researching security of Wireless Mesh Networks, but still they lack efficient and scalable security solutions due to their dynamic and distributed nature of the system. The security factor can be explored with various key challenges includes secure routing, authentication, access control and authorization, key management and intrusion detection. Further, none of the existing key management based techniques are suitable for wireless mesh networks as they are inefficient on an arbitrary or unknown network topology, or not tolerant to a changing network topology or link failures.
In my research I focus on secure routing in WMN as the other aspects have been talked about in the literature and there has been urgent necessity of further exploration of secure routing in WMN.

3. Emerging concept, from nature to communication engineering

A great increase in studying biologically inspired systems, specifically the rising curiosity in Human Immune System (HIS) stimulates the applicability of HIS concepts in WMN security. Some of the attractive features analogous to the features of WMN include adaptability, distributability, diversity, autonomy and dynamic coverage. Moreover, secure

routing functions can be mocked-up by analysing the reaction process of Human Immune System (HIS) against a foreign material.

4. Immune system

4.1 Human Immune System

Human Immune System is an extremely complex collection of cells and organs that work together to defend the body against foreign attacks. It has the capability of recognizing different enemies. Once the immune cells receive the distress they go through changes and start producing chemicals which regulates the growth and the behavior of the cells (NIH 2003). A healthy immune system is capable of distinguishing between self cells and non-self cells. The fig. 6. shows the structure of the HIS.

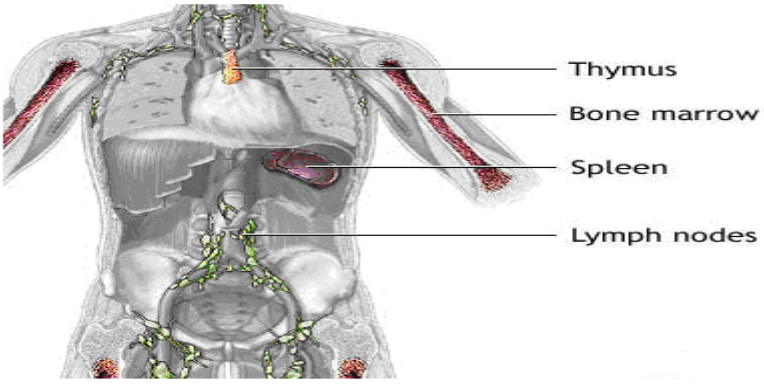


Fig. 6. Structure of HIS (Source Mediline)

4.2 How does Immune System respond to Antigen?

Human Immune System plays major role in protecting human body against pathogens and microbes. The large collection of cells in HIS operates autonomously and creates series of events leading to the destruction of pathogens. Immune cells can be broadly categorized in to two groups namely detectors and effectors (Mahira et al., 2007), detectors identify pathogens, and effectors neutralize them. Moreover, two kinds of immune responses are induced by the Immune system. They are innate response and adaptive response. During innate immune response process pathogens in the body are detected by phagocyte and antibodies are produced by the adaptive immune response to recognize specific pathogens. The lymphocytes that match antigen propagate by cloning and subsequently differentiate into B-cells, which generate antibodies, and T-cells, which destroy infected cells and activate other cells in the immune system (Mahira et al., 2007).

4.3 Artificial Immune System Models

There are four Artificial Immune System models discussed in the literature. They are negative selection model, clonal selection model, immune network model and danger model.

4.3.1 Negative Selection Model

This was introduced by Forrest in 1994 as a conceptual model of biological negative selection. Change detection is the basis of this algorithm and the generated detectors are intended to detect self strings which have changed from an ascertain norm.
In this process, firstly, set of self strings and a set of random strings are created. Secondly matching function is defined for random strings which do not strongly match as self string. This process iterates until detector strings are obtained.

4.3.2 Clonal Selection model

The basis for the clonal selection algorithm is the natural B-cell mechanism (Mahira et al., 2007). When the receptors of immature B-cells in the blood match to an antigen they propagate rapidly and modify to facilitate better matching. The B-cells with better matching proliferate continuously until the best matching B-cells are produced (Mahira et al., 2007) . Leandro N. de Castro and Fernando J. Von Zuben proposed natural clonal selection based algorithm, called CLONALG .This is a representation of computational implementation of clonal selection and affinity maturation principles accountable for behaviour of B-cells during adaptive immune responses .Fig.7. shows the clonal selection process.

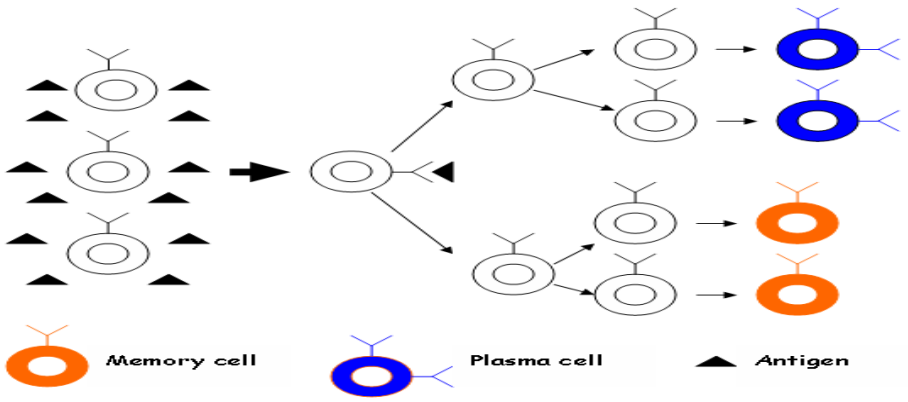


Fig. 7. Clonal Selection Process [Source (Mahira et al., 2007)]

4.3.3 Immune Network Model

The immune network theory suggests that the immune system has a dynamic behaviour even in the absence of external stimuli. Further the immune cells and molecules are capable of recognizing each other, which provides the system with an eigen behaviour that is independent of alien stimulation. The recognition of antigen by an antibody (cell receptor)

leads to network activation, whereas the recognition of an idiotope by another antibody leads to network suppression. According to the immune network theory, the receptor molecules contained in the surface of the immune cells present idiotopes, and these idiotopes are displayed in and/or around the same portions of the receptors that recognize non-self antigens.

4.3.4 The Danger Theory

Classical immunology depends on the concept of "self" and "non-self" cells distinction. An immune response is triggered when the body encounters something "non-self", Matzinger pointed out that there is a bias in differentiating self and non-self cells because the HIS does not respond to useful bacteria in the food or air (Matzinger, 2002) . On the other hand, the central theme of Danger theory is that the immune system responds to danger but not to "non-self". The motive for this is that there is no need to attack everything that is foreign. This concept is very practical in WMN environment. The danger can be measured in terms of distress signal sent out by unexpectedly dying nodes/devices in the network. Fig.8 shows how an immune response can be pictured according to the danger theory. A cell that is in distress sends out danger signals and form danger zone around itself, then antigens in the neighborhood are captured by Antigen-presenting cells and they travel to the local lymph node and present the antigens to the lymphocytes. B-cells, which are within the danger zone, get stimulated to produce antibodies that match the antigens and to traverse the clonal expansion process. Those which do not match or are too far away do not get stimulated.

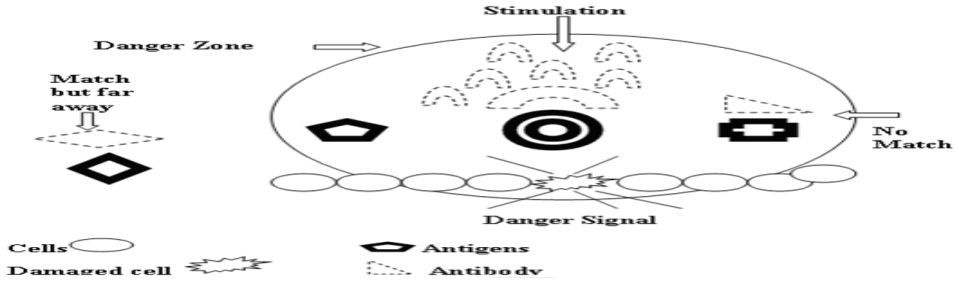


Fig. 8.Model of the Danger Theory [source (Mahira et al., 2008)]

Further, self and non-self classification against danger and non-danger distinction is illustrated in fig.9.

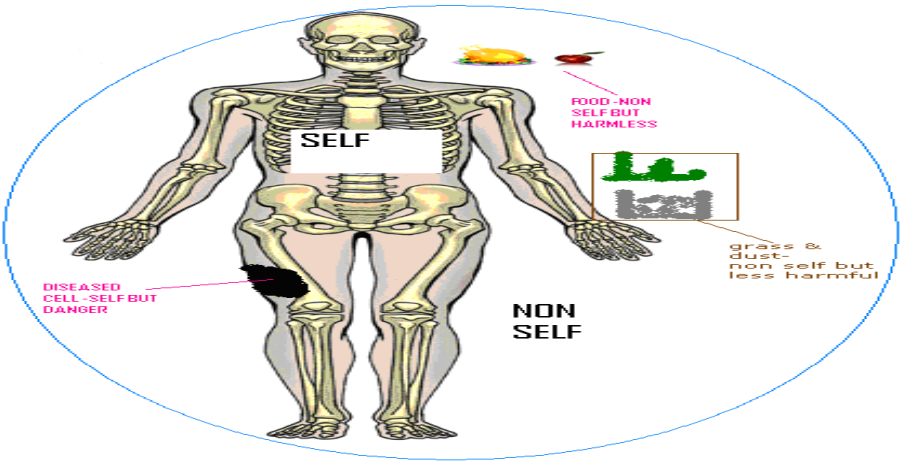


Fig. 9. Self and non-self classification against danger and non-danger distinction

5. Mapping of HIS elements on to WMN elements

As the first step in implementing HIS inspired secure routing, mapping of Human Immune System Elements onto Wireless Mesh Network has been carried out. The table 1 shows the mapping of HIS elements on to WMN devices.

HIS	WMN
Body	The entire WMN system
Self-Cells	Well behaved network resource nodes
Non-Self Cells	Corrupted or well-behaving but unauthorized nodes inside the network or any external input either friendly or malicious. Inactive or non participating nodes
Antigen	Possible cause of interruption or anomalies or danger to the network
Antibody	Recovery or protection actions for the node in danger possibly caused by antigen
Cytokines	Error messages or danger signals or events communicated between nodes

Table 1. Mapping of HIS elements on to WMN devices

6. Clustering methods

The main goal of clustering is to analyse and reduce the amount of data to work with by grouping related data elements together. Different clustering techniques are used in various application scenarios, for example clustering mechanisms used in web data grouping, spatial data clustering, in Biology and in marketing research. Taxonomy of clustering approaches is given in (Jain et al., 1999). The author has given an extremely detailed description of clustering mechanisms in his paper (Pavel). Partitional clustering and hierarchical clustering are the basic types of clustering mechanisms considered in (Ugur), (Samuel & Nick Theodosopoulos, 2006). Hierarchical algorithms can be performed as

bottom up approach or top down approach. In this approach clusters are formed at the beginning of the process and then new clusters are produced based on the relationship of the data elements within the data set. On the other hand partitional algorithms form the clusters only once during the process. Then each element of the dataset is analysed and placed within the corresponding cluster.

7. Self-Organizing Map (SOM)

Self organizing map (SOM) is a computationally efficient neural network which is introduced by der Malsburg in 1973 and subsequently by Kohonen in 1982. SOM is also called as Self organizing feature maps (SOFM), topographic map and Kohonen feature maps. In the simplest form, SOM is a set of cells organised in different layers and each cell represents a piece of data which is usually a vector. Further, SOM admits N-dimensional input vectors and maps them to the Kohonen layer, in which neurons are usually arranged in a 1, 2 or 3 dimensional grid representing the feature space. Such a grid characterizes the topological properties of neurons rather than exact geometric locations. Fig.10 shows the basic structure of SOM and this architecture maps high dimensional data on to a two dimensional rectangular grid of output layer and the input layer is fully connected with the output layer and the output neurons have lateral connections to their neighbours.

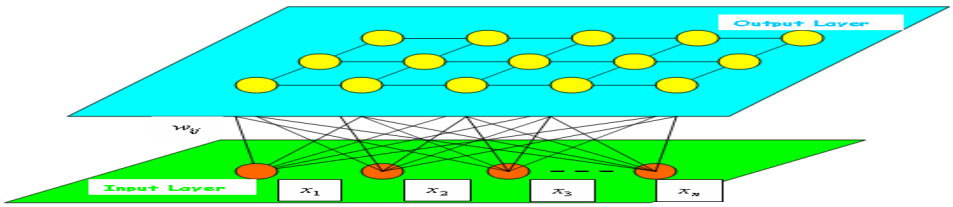


Fig. 10. The basic structure of SOM

In comparison with other commonly used clustering techniques, SOM has been broadly applied in data analysis and tremendous clustering and classification performance have been reported (Hashemi et al., 2005). Moreover, SOM provides comprehensible graphical illustration of the input data patterns, which has been applied to identify new vulnerability patterns while new threats or attacks amplify. The Growing Self-Organizing Map (GSOM) is a division of SOM with the growing map size. The motivation behind the GSOM is that the exact topology and the size of the map often have a large impact on the training process of the SOM and the map is determined by the statistical regularity not by the programmer. A detailed description of GSOM is given in (Hashemi et al., 2005).

8. Classification of Dangers using SOM

Mathematical Modelling

It is assumed that there is Q number of input patterns and the input vectors are of dimension N. Therefore the output for neuron j, from input pattern i, is denoted by $y_{i \to j}$ and depicted in Fig.11: Then the winning output neuron j, is determined by selecting the

output neuron with the best weight vector that matches the input vector. This is achieved by calculating the distance between x_i and w_j. Where x_i and w_j are derived from the equations (1) and (2) and the distance $d_{i \to j}$ is manipulated using the equation (3).

$$\mathbf{x}_i = \begin{pmatrix} x_{i,1} \\ \\ x_{i,N} \end{pmatrix} \tag{1}$$

$$\mathbf{w}_j = \begin{pmatrix} w_{j,1} \\ \\ w_{j,N} \end{pmatrix} \tag{2}$$

$$d_{i \to j : x_i \to w_j} = \left[\sum \left(x_{i,k} - w_{j,k} \right) \right]^{1/2} \tag{3}$$

Moreover, x_{Total} and $y_{i \to j}$ are calculated using the formula (4) and (5) respectively.

$$\mathbf{x}_{Total} = \begin{bmatrix} \mathbf{x}_1 \\ \downarrow \\ \mathbf{x}_Q \end{bmatrix} = \begin{bmatrix} \left(x_{1,1} \rightarrow x_{1,N} \right)^T \\ . \\ \left(x_{Q,1} \rightarrow x_{Q,N} \right)^T \end{bmatrix} \tag{4}$$

$$y_{i \to j} = \sum_{k=1}^{N} w_{j,k} \; x_{i,k} \tag{5}$$

where,

i = 1...Q (Q = number of input patterns)
j = 1....M (M = dimension of output neurons)
k = 1.....N (N = dimension of input vectors)

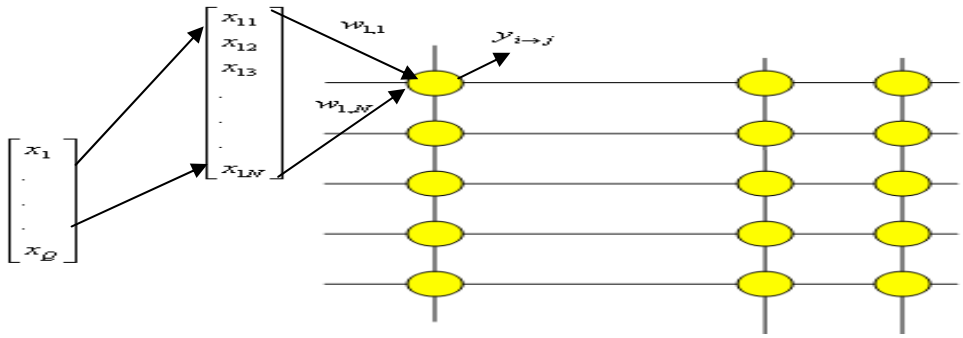


Fig. 11. The output for neuron j, from input pattern i

In addition, adjust weights of the winner neuron towards the input is given by the equation (6).

$$\mathbf{w}_j(t+1) = \begin{cases} \mathbf{w}_j(t) + \eta(t)\left[\mathbf{x}_i(t) - \mathbf{w}_j(t)\right] & j \in \Lambda_{winner}(t) \\ \mathbf{w}_j(t) & otherwise \end{cases} \tag{6}$$

Where, $\eta(t)$ is the decrease learning rate which is in a linear fashion or exponential decay; reduce the neighbourhood $\Lambda_{winner,neigh}(t)$ to update the weights of the neighbours,

$$\mathbf{w}_j(t+1) = \mathbf{w}_j(t) + \eta(t)\ \Lambda_{winner,neigh}(t)\ \left[\mathbf{x}_i(t) - \mathbf{w}_j(t)\right] \tag{7}$$

$\wedge_{winner,neigh}(t)$ is defined as follows,

$$\Lambda_{winner,neigh}(t) = e^{\left(\frac{-\left\|r_{neigh} - r_{winner}\right\|^2}{2\sigma^2}\right)} \tag{8}$$

Then the size of neighbourhood decreases over time to stabilize mapping, where σ represents width of neighbourhood function which is an exponential decay.
Based on these theoretical analyses, experiment was carried out and reported in section 9.

9. Experiment

In order to evaluate the performance of the proposed approach with DT, an experiment is conducted based on the process described in the previous sections. Self-Organizing Maps (SOMs) are applied as a danger level classifier in the experiments. To simplify the experiments, the input of SOMs in this simulation are limited to the following three categories: (1) Changes of Self-nonself information flow during a time interval, e.g., attack

information and node failures; (2) Network traffic conditions, e.g., number of transmitted packets, bit error rate (BER) and packet delivery ratio (PDR). (3) Resource conditions, e.g., remaining bandwidth or memory, Link capacities.

9.1 Distributed Internet Traffic Generator (D-ITG)

The Distributed Traffic Generator (D-ITG) is used to generate artificial traffic data in order to train and test SOM network. The TCP traffic with Poisson distribution is adopted where packet size is 512 bytes, and average 1000 packets/sec. Beside, UDP traffic is also considered in the dataset. The bounds to identifying different level of traffic load are setup and the table 2 shows the details. These traffic conditions relates to the classification of danger levels.

Testbed Elements	Descriptions	Details
D-ITG	{UDP, TCP}	Poisson Distributions for TCP
	{Packet Size} byte	{64, 128, 256, 512, 1024, 1500}
	{Traffic Load}	Low ($\leq 1.5Mbps$) Medium ($\leq 6Mbps$) High ($\leq 8Mbps$)

Table 2. Parameter Elements

Compared to the higher dimensionality of input data, the output of SOMs is a lower dimension space. When the network is fully trained, it is ready to get the data clustered on the map demonstrated in the Figure 14. After some tests by using the testing data, the 4 classified zones defined in the figure are found. In the simulation, a 2-dimensional space split into 4 classified zones, such as High danger, Medium danger, Low danger and Unacceptable zone, are the output of SOMs output neurons.

Figure 12 shows the process of training SOMs and testing SOMs with the testing data, which is part of the training dataset generated by the Distributed Traffic Generator - D-ITG and input dataset which is generated by Monte Carlo Simulation where random number generator produce all the dataset. Table3 shows the parameter setting of the SOMs for identifying danger levels. The following Figure 14 shows the simulation result. Input data is classified via SOM and visualized results are represented.

Number of Input Layer	3
Row of Output Layer	20
Column of Output Layer	20
Topology	Rectangular
Learning efficiency	0.9
Iteration Number	2000
Error_limit	5E-12

Table 3. Parameters of SOM for danger levels

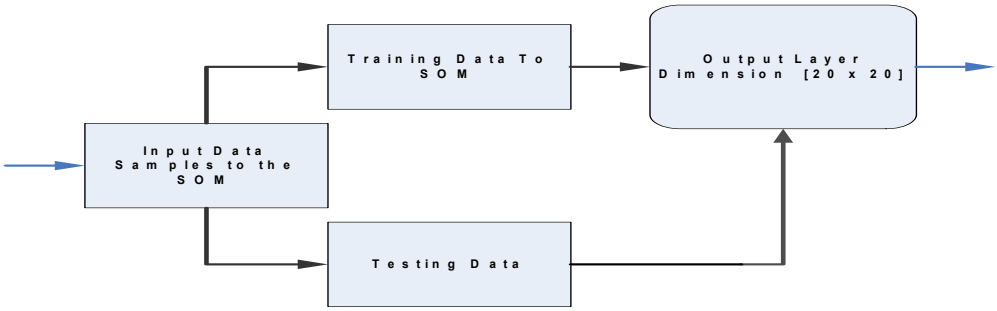


Fig. 12. Simulation Structure

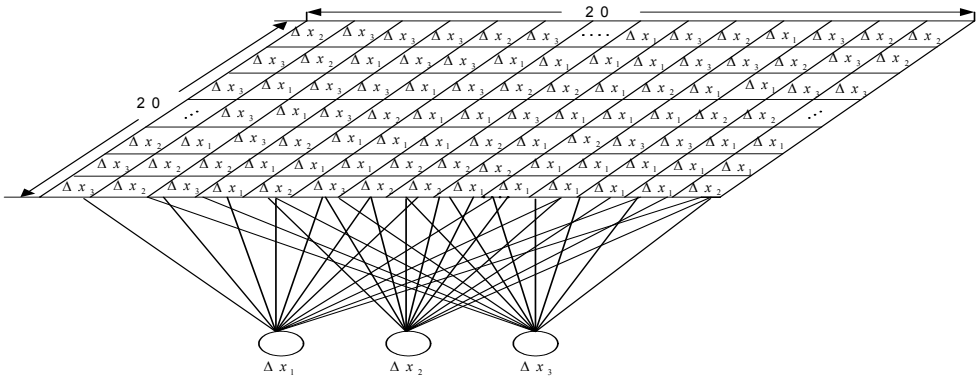


Fig. 13. Structure of SOM in the simulation

9.2 Simulation Results

According to the classified danger information from SOM, the methods and algorithms defined in this chapter will be invoked for secure routing protocol in the research. The simulation results of the algorithm are shown in Figure 14, as shown, the input data vectors involving three types of inputs are classified into 4 zones: (1) High Danger Level, (2) Medium Danger Level, (3) Low Danger Level (4) Unaccepted Danger Level. The darkest color shows zone (4). The lightest color represents zone (1) The other two zones (2) and (3) are depicted by the blue color. The lighter blue color zone represents zone (2). These zones stand for the danger levels.

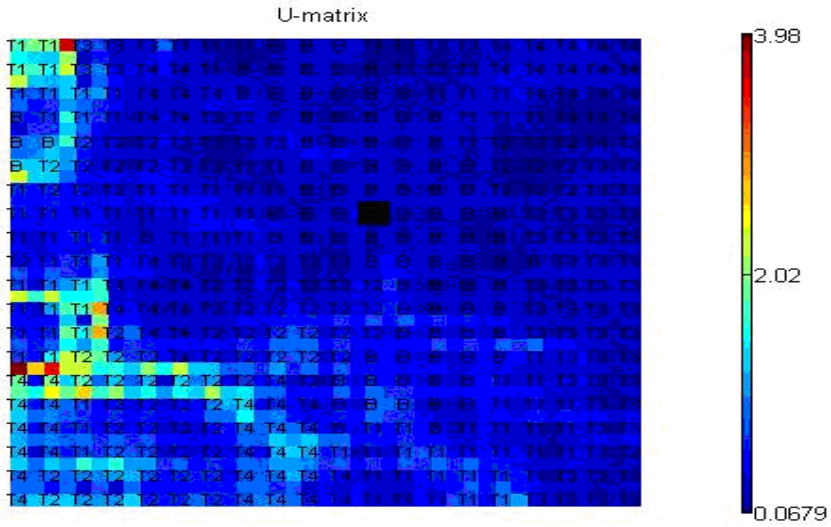


Fig. 14. Simulation Result

Acknowledgment

The authors wish to thank Zenon Chaczko and Frank Chiang for their contribution in preparing the paper, "Mahira Atham Lebbe, Johnson I Agbinya, Zenon Chaczko and Frank Chiang, Self-Organized Classification of Dangers for Secure Wireless Mesh Networks, 2007 Australasian Telecommunication Networks and Applications Conference, December 2nd – 5th 2007, Christchurch, New Zealand"

10. References

AK Jain , MN Murty and PJ Flynn (1999), Data Clustering: A Review, *in ACM Computing Surveys*, pp 264-323 Vol. 31, No. 3, September 1999.

Hashemi, R.R., M. Bahar, and S. De Agostino (2005). An extended self-organizing map (ESOM) for hierarchical clustering, *proceedings of IEEE International Conference on Systems, Man and Cybernetics*, pp 2856- 2860 Vol. 3, ISBN: 0-7803-9298-1, *Hilton, Las Vegas, August, 2005, IEEE, Nevada, USA*.

Hidenori Aoki,Shinji Takeda, Kengo Yagyu and Akira Yamada (2006) , IEEE 802.11s Wireless LAN Mesh Network technology, *NTT DoCoMo technical Journal*, Vol.8.No.2, pp 13-21.

Mahira Atham Lebbe, Johnson I Agbinya, Zenon Chaczko and Frank Chiang (2007), Self-Organized Classification of Dangers for Secure Wireless Mesh Networks, *proceedings of Australasian Telecommunication Networks and Applications Conference 2007*, pp 322-327, ISBN: 978-1-4244-1557-1, Christchurch December 2007,IEEE , New Zealand.

Mahira Atham Lebbe (Mahira M.Mowjoon), Johnson I Agbinya, Zenon Chaczko, Robin Braun (2008). Artificial Immune System inspired danger modelling in Wireless Mesh Networks, *proceedings of International Conference on Computer and Communication Engineering*, pp 984-988, ISBN: 978-1-4244-1691-2 , Kuala Lumpur , May 2008, IEEE, Malaysia.

Matzinger P (2002), The danger model: A renewed sense of self, *Science Magazine*, vol. 296, no. 5566, pp. 301–305.

Pavel Berkhin, Survey of Clustering Data Mining Techniques, *in Accrue Software, Inc*., pp 1-56, 1045 Forest Knoll Dr. San Jose, CA 95129, http://www.ee.ucr.edu/~barth/EE242/clustering_survey.pdf, available and validated on 31.03.2009

Samuel Sambasivam and Nick Theodosopoulos (2006), Advanced Data Clustering Methods of Mining Web Documents, *proceedings of Issues in Informing Science and Information Technology*, pp 563-579 Volume 3, Salford, June 2006, England.

Ugur Halici, Artificial Neural Networks, *In: Artificial Neural networks*, pp126-138, EE543 Lecture Notes, METU EEE Ankara.

Understanding the Immune System, How It Works, *US department of Health and Human services, National Institute of Health and National Institute of Allergy and infectious diseases National Cancer Institute*, NIH publication no.03-5423, September 2003,

http://www.niaid.nih.gov/Publications/immune/the_immune_system.pdf available and validated on 05.03.2009

Mediline plus Medical Encyclopedia: Immune system structures, http://www.nlm.nih.gov/medlineplus/ency/imagepages/8932.htm, available and validated on 05.03.2009

A Knowledge Acquisition Method of Judgment Rules for Spam E-mail by using Self Organizing Map and Automatically Defined Groups by Genetic Programming

Takumi ICHIMURA, Kazuya MERA and Akira HARA
Hiroshima City University
Japan

1. Introduction

Recently, the Internet has been a basis of our cultural life. Web provides a virtual huge space of information, where an emotional experience, a political idea, a cultural custom, and the advice of manners of music, the business, the arts, photographs, and literatures, etc. are digitalized at a low price and are shared for our current culture. One of major other tools gives E-mail communication which propagates not only letters from person to person, but a means of advertisement. However, E-mail addresses on the web page are gained and the virus is appended to E-mails, and the attacks for acquiring information on user's personal computer (called BOT) have been spreaded. Their accumulated E-mail addresses collected in such a way will be valuable for advertising agents. But, their E-mails are called "Spamming" and the Spamming becomes a one of the social issues.
Spamming is the abuse of electronic messaging systems to send unsolicited bulk messages or to promote products or services, which are almost universally undesired. Spamming is economically viable because advertisers have no operating costs beyond the management of their mailing lists. The sender cannot be specified, because the sender of Spamming has only temporary E-mail address and the reply of them is not reached to the original sender. Therefore, undesired E-mails to us have been increased everyday, so that, it is not easy to read an important E-mail.
In order to avoid Spam E-mails, we must build the filtering system which can judge whether the received E-mail is a Spam E-mail or not. The SpamAssassin(SpamAssassin) is the open source software and is a mail filter which attempts to identify a Spam E-mail using various pattern match methods including text analysis, Bayesian filtering, DNS block lists, and collaborative filtering databases. There are the predefined rules for each filtering method to detect a Spam E-mail. The agreement degree for each rule is scored and if the total score is larger than the threshold value, the given E-mail is judged as Spam. Although each score in the rule of SpamAssassin is low, there is a few cases in which total score becomes high. Moreover, even if the message is judged as Spam, the different rules for each person are

required according to the personal environment such as work style, because a content of received message will be different.
In this paper, we propose a classification method for Spam E-mail based on the results of SpamAssassin, which is the open source software to identify spam signatures. This method can learn patterns of Spam E-mails and Ham ones and correctly recognizes them. First, the method divides E-mails into some categories by Self-Organizing Map (SOM) (Kohonen, 1995) and extracts the adequate judgment rules by Automatically Defined Groups(ADG) (Hara, 1999), even if the judgment results by SpamAssassin are wrong.
The SOM is developed by Kohonen and is a topology-preserving map because there is a topological structure imposed on the nodes in the network. A topological map is a simple mapping that preserves neighborhood relations. The SOM is an algorithm used to visualize and interpret large high-dimensional data sets.
The ADG is a new method that united Genetic Programming (GP) with cooperative problem solving by multiple agents. By using this method, we had developed the rule extraction system from database (Hara, 2004, 2005, 2008). In this system, two or more rules hidden in the database and respective rules' importance can be acquired by cooperation of agents.
In order to verify the effectiveness of our proposed method, about 3,000 Spam and Ham E-mails are examined. The SpamAssassin works in the Linux Server where Postfix(Frederick) operates as Mail Transfer Agent (MTA) and the interface between MTA and content checkers is amavisd-new (Amavisd-new). Section 2 describes the operation of SpamAssassin and Postfix with an interface called amavisd-new. Section 3 and 4 explain the mechanism about SOM and ADG. Section 5 described the experimental results. Section 6 gives the conclusive discussion.

2. SpamAssassin under MTA

The SpamAssassin is a flexible and powerful set of Perl programs which score the agreement degree from multiple types of checks to determine whether a given E-mail is Spam. Because the SpamAssassin has no function to receive or send an E-mail, it must operate with MTA such as Postfix. Fig.1 shows the operation between Postfix and SpamAssassin through Amavisd-new. The clamav in Fig.1 is the Clam AntiVirus (Clam AntiVirus) which is a GPL anti-virus toolkit for UNIX, designed especially for E-mail scanning on mail gateways.
The SpamAssassin has the following features:
- Header tests
- Body phrase tests
- Bayesian filtering
- Check of E-mail address of blacklist/whitelist automatically
- Check of E-mail address of blacklist/whitelist manually
- Tests by using Collaborative Spam identification databases
- Tests by using DNS Blocklists
- Tests of Character sets and locales

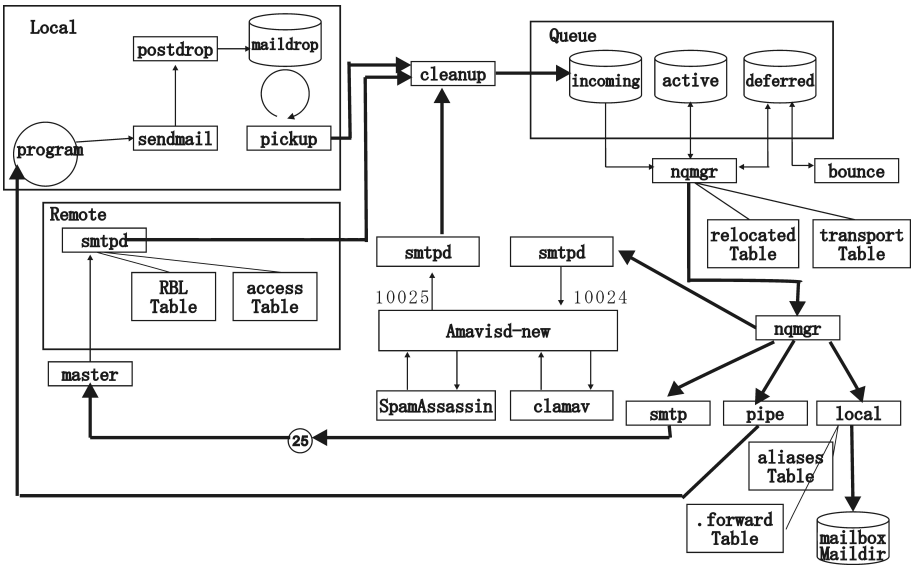


Fig.1 A flow of MTA and SpamAssassin

Even if any one of these tests may not identify a Spam or a Ham correctly, it is not easy to make the correct judgment by only their combined score. For example, Fig.2, Fig.3, and Fig.4 show the headers of E-mails which are judged as Spams through the SpamAssassin.

```
X-Spam-Flag: YES
X-Spam-Score: 57.834
X-Spam-Level: ************************************************************
X-Spam-Status: Yes, score=57.834 tagged_above=2 required=6.31
tests=[AWL=10.800, BAYES_99=7.5, CHINANET=1, CNCGROUP=1.5, CNCJP=1.5,
DNSFRMRFCPST99=1.5, DNS_FROM_RFC_POST=0.1, DYN_DNSFRMRFCPST=3.5,
INVALIDYAHOOJP=1, INVYJP_DYN=3.5, ISO2022JP_BODY=-0.1,
NOTINCONTENTTYPE=0.2, QENCPTR1=0.2, REVDNSUNKNOWN=0.2,
SPF_SOFTFAIL=1.384, SURBL99=3.5, URIBL_AB_SURBL=3.812,
URIBL_JP_SURBL=1, URIBL_OB_SURBL=0.1, URIBL_SC_SURBL=4.498,
URIBL_WS_SURBL=2.14, URLBL_RBLJP=1.5, URLRBLJP99=2, URLRBLJP_DYN=5.5]
```

Fig.2 SpamAssasin Example 1 (SPAM)

```
X-Spam-Flag: YES
X-Spam-Score: 8.916
X-Spam-Level: ********
X-Spam-Status: Yes, score=8.916 tagged_above=2 required=6.31
tests=[AWL=-2.684, BAYES_99=7.5, DNSFRMRFCABS99=0.2,
DNS_FROM_RFC_ABUSE=0.1, NOTINCONTENTTYPE=0.2, SURBL99=3.5,
URIBL_OB_SURBL=0.1]
```

Fig.3 SpamAssasin Example 2 (HAM)

```
X-Spam-Flag: YES
X-Spam-Score: 7.814
X-Spam-Level: *******
X-Spam-Status: Yes, score=7.814 tagged_above=2 required=6.31
tests=[AWL=3.913, BAYES_50=0.001, CLICK_JP=1, CONTENT_TYPE_PRESENT=-0.1,
DNS_FROM_RFC_ABUSE=0.1, GEKIYASU=0.5, HAISHINTEISHI=0.3,
HIMITSUNO=0.1, HTML_MESSAGE=1, ISO2022JP_BODY=-0.1,
ISO2022JP_CHARSET=-0.1, MIME_HTML_ONLY=0.4, MURYOU=0.2, QENCPTR1=0.2,
RENRAKU=0.2, SUBJECT_ENCODED_TWICE=0.1, X_MAILER_PRESENT=0.1]
```

Fig.4 SpamAssasin Example 3 (HAM)

Fig.2 is the message judged at the score of 57.834 as Spam E-mail. However, the scores of examples as shown in Fig.3, and Fig.4 were 8.916 and 7.814 respectively which were judged as Spam, but the messages are Ham. Especially the score of "BAYES_99"' was 7.5 in Fig.2, and the result shows that classification capability of Bayesian filtering is not high. In this paper, 3,007 E-mails are accumulated in the database, where there are 2,913 Spams and 94 Hams misjudged as Spams.

3. Self Organizing Map(SOM)

The basic SOM can be visualized as a sheet-like neural network array as shown in Fig.5, the cells (or nodes) of which become specifically tuned to various input signal patterns or classes of patterns in an orderly fashion. The learning process is competitive and unsupervised, which means that no teacher is required to define the correct output for an input. Only one map node called a winner node at a time is activated corresponding to each input. The map consists of a regular grid of processing units. A model of some multidimensional observations, eventually a vector consisting of features, is associated with each unit. The map attempts to represent all the available observations with optimal accuracy using a restricted set of models. At the same time the models become ordered on the grid so that similar models are close to each other and dissimilar models are far from each other.

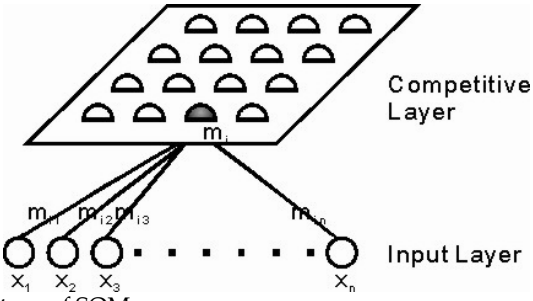


Fig.5 A basic architecture of SOM

Fitting of the model vectors is usually carried out by a sequential regression process. The n is the dimensioned number of input signals. An input vector $\mathbf{x}$ is compared with all the model vectors $\mathbf{m}_i(t)$. The best-match unit on the map is identified. The unit is called the winner.

For each sample $\mathbf{x} = \{x_1, x_2, \cdots, x_n\}$, first the winner index c (best match) is identified by the condition

$$\forall i, \|\mathbf{x} - \mathbf{m}_c\| \leq \|\mathbf{x} - \mathbf{m}_i\|$$

After that, all model vectors or a subset of them that belong to nodes centered around node c are updated at time t as

$$\mathbf{m}_i(t+1) = \mathbf{m}_i(t) + h_{ci}(\mathbf{x}(t) - \mathbf{m}_i(t)) \quad for \forall i \in N_c(t)$$
$$\mathbf{m}_i(t+1) = \mathbf{m}_i(t) \quad otherwise$$

Here $h_{ci(x)}$ is the neighborhood function, a decreasing function of the distance between the i th and c th nodes on the map grid. The $N_c(t)$ specifies the neighborhood around the winner in the map array. This regression is usually reiterated over the available samples.
In this paper, we tried to classify the 3,007 E-mails into Spams and Hams by SOM in order to obtain the visual distribution intuitively.

4. Rule Extraction by ADG

4.1 Automatically Defined Groups

In the domain of data processing, to cluster the enormous data and to extract common characteristic from each clustered data are important for knowledge acquisition. In order to accomplish this task, we adopt a multi-agent approach, in which agents compete with one another for their share of the data, and each agent generates a rule for the assigned data; the former corresponds to the clustering of data, and the latter corresponds to the rule extraction in each cluster. As a result, all rules are extracted by multi-agent cooperation. However, we do not know how many rules subsist in given data and how data should be allotted to each agent. Moreover, as we prepare abundant agents, the number of tree structural program increases in an individual. Therefore, the search performance declines.

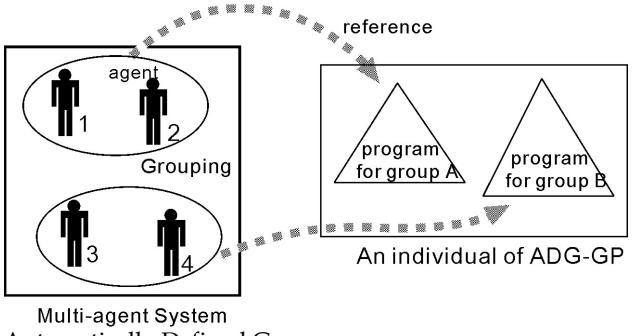


Fig.6 Concept of Automatically Defined Groups

In order to solve these problems, we have proposed an improved GP method, Automatically Defined Groups (ADG). The method optimizes both the grouping of agents

and the program of each group in the process of evolution. By grouping multiple agents, we can prevent the increases of search space and perform an efficient optimization. Moreover, we can easily analyze the behavior of agents. Respective groups play different roles from one another for cooperative problem solving. The acquired group structure is utilized for understanding how many roles are needed and which agents have the same role. That is, the following three points are automatically acquired by using ADG.

- How many groups (roles) are required to solve the problem?
- Which group does each agent belong to?
- What is the program of each group?

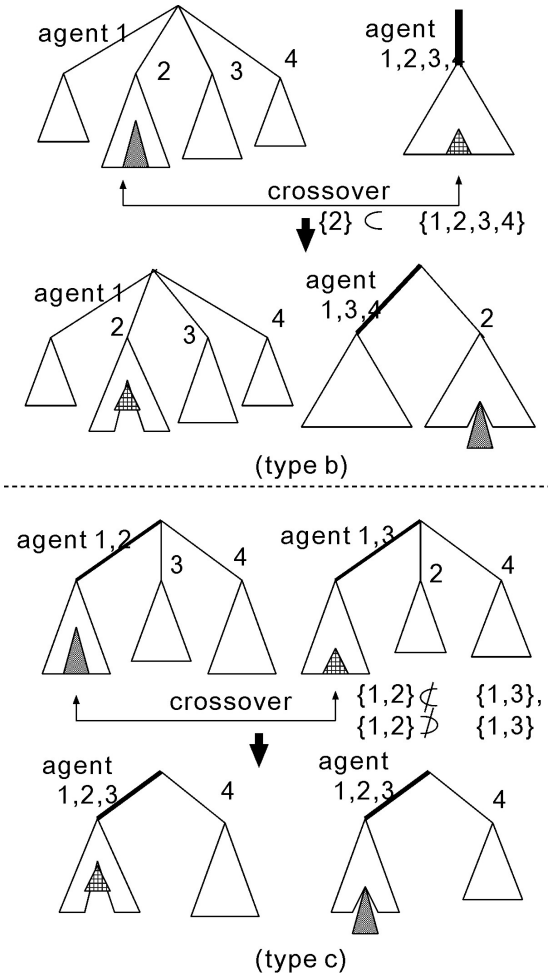


Fig.7 Examples of crossover

In ADG, each individual consists of the predefined number of agents. One GP individual maintains multiple trees, each of which functions as a specialized program for a distinct

group as shown in Fig.6. We define a group as the set of agents referring to the same tree for the determination of their actions. All agents belonging to the same group use the same program.

Generating an initial population, agents in each GP individual are divided into several groups at random. Crossover operations are restricted to corresponding tree pairs. For example, a tree referred to by an agent 1 in an individual breeds with a tree referred to by an agent 1 in another individual. In ADG, we also consider the sets of agents that refer to the trees used for the crossover. The group structure is optimized by dividing or unifying the groups according to the inclusion relationship of the sets.

The concrete processes are as follows: We arbitrarily choose an agent for two parental individuals. A tree referred to by the agent in each individual is used for crossover. We use T and T' as expressions of these trees, respectively. In each parental individual, we decide a set $A(T)$, the set of agents that refer to the selected tree T. When we perform a crossover operation on trees T and T', there are the following three cases.

(a) If the relationship of the sets is $A(T) = A(T')$, the structure of each individual is unchanged.

(b) If the relationship of the sets is $A(T) \supset A(T')$, the division of groups takes place in the individual with T, so that the only tree referred to by the agents in $A(T) \cap A(T')$ can be used for crossover. The individual which maintains T' is unchanged. Fig.7 (type b) indicates an example of this type of crossover.

(c) If the relationship of the sets is $A(T') \not\subset A(T)$ and $A(T) \not\subset A(T')$, the unification of groups takes place in both individuals so that the agents in $A(T) \cup A(T')$ can refer to an identical tree. Fig.7 (type c) shows an example of this crossover.

We expect that the search works efficiently and the adequate group structure is acquired by using this method.

4.2 Rule Extraction from classified data

In some kinds of databases, each data is classified into positive or negative case (or more than two categories). It is an important task to extract characteristics for a target class. However, even if data belong to the same class, all the data in the class do not necessarily have the same characteristic. A part of data set might show a different characteristic. It is possible to apply ADG to rule extraction from such classified data. In ADG, multiple tree structural rules are generated evolutionally, and each rule represents the characteristic of a subset in the same class data. Fig.8 shows a concept of rule extraction using ADG. Each agent group extracts a rule for the divided subset. The rules acquired by multiple groups can cover all the data in the target class.

We describe the detail of rule extraction from classified data. Here, the rules whether input data is classified into positive cases are extracted. In order to judge whether each data is regarded as positive case, we will find logical expressions such that the only positive data should satisfy.

Multiple trees in an individual of ADG represent the respective logical expressions. Each data in the training set is input to all trees in the individual. Then, calculations are performed to determine whether the data satisfy each logical expression. The input data is regarded as positive case if one or more logical expressions in the individual returns true. In

contrast, the input data is regarded as negative case if all logical expressions in the individual return false.

The concept of each agent's load arises from the viewpoint of cooperative problem solving by multiple agents. The load is calculated from the adopted frequency of each group's rule and the number of agents in each group. The adopted frequency of each rule is counted when the rule successfully returns true for each positive data. If multiple trees return true for a positive data, the tree with more agents is adopted. When the agent a belongs to the group g, the load of the agent w_a is defined as follows:

$$w_a = \frac{f_g}{n_{agent}^g}$$

where n_{agent}^g represents the number of agents which belong to the group g, and f_g represents the adopted frequency of g. By balancing every agent's load, more agents are allotted to the group that has a greater frequency of adoption. On the other hand, the number of agents in the less adopted group becomes small. Therefore, the number of agents of respective rules indicates how general each rule is for judgment of the class. Moreover, when some cases are judged to be true through a mistake of a rule, it is thought that the number of agents who support the rule should be small. To satisfy the requirements mentioned above, we will maximize the fitness f defined as follows:

$$f = -\frac{miss\_target\_data}{N_{Positive}} - \alpha\frac{misrecognition}{N_{Negative}} - \beta\frac{\sum_{N_{Negative}} fault\_agent}{misrecognition \times N_{agent}} - \delta V_w$$

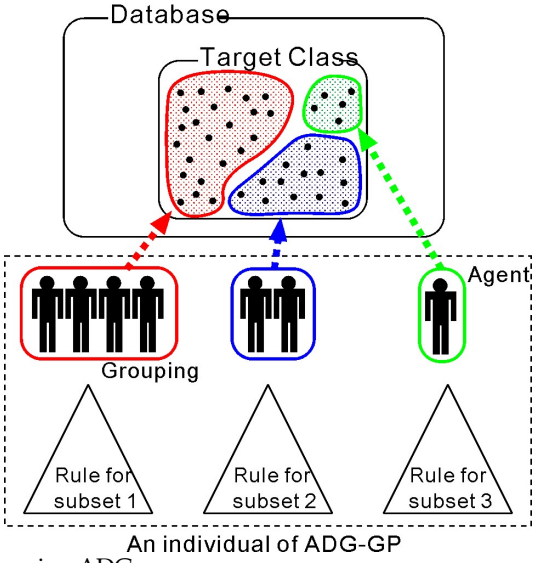


Fig. 8. Rule extraction using ADG

In this equation, $N_{Positive}$ and $N_{Negative}$ represent the number of positive cases and negative cases in database respectively. $miss\_target\_data$ is the number of missing data in the target positive data that should have been judged to be true. *misrecognition* is the number of mistakes through which negative data is regarded as positive case. When the rule returns true for negative data, $fault\_agent$ is the number of agents who support the wrong rule in each data. So, the third term represents the average rate of agents who support the wrong rules when misrecognition happens. By this term, the allotment of agents to a rule with more misrecognition will be restrained. V_w is the variance of every agent's load. By the fourth term, load balancing of agents will be achieved.
By evolution, one of the multiple trees learns to return true for positive cases, and all trees learn to return false for negative cases. Moreover, agents are allotted to respective rules according to the adopted frequency, and the allotment to a rule with more misrecognition is restrained. Therefore, the rule with more agents is the typical and reliable classification rule, and the rule with less agents is an exceptional rule for the rare case. This method is applied to the medical data and the effectiveness is verified (Hara, 2004, 2005).

5. Experimental Results

5.1 Learning for all cases and Extraction of rules(ichimura, 2007)

The map in the trained SOM as shown in Fig. 9 creates the distribution of Spam and Ham E-mails intuitively. The numerous label indicate the index of E-mails and each index is a category of E-mails which is classified the characters of judgment result by SpamAssassin. Each category usually has two or more classified E-mails. However, a category consisted of not only the message judged as Spam but the Ham E-mail misjudged as Spam. If the message is classified into this category, we have to reexamine the result of the classification, because the E-mail may be a Ham. In such a case, we will try to apply the sophisticate classification algorithm as shown in (ichimura, 2004; Ichimura, 2005; Yamaguchi 2008a; Yamaguchi 2008b).
Moreover, to make the process of reexamination automatically, we applied the extraction rules of finding a Ham misjudged as Spam by ADG. In ADG, two kinds of node are used for rule representation; function node and terminal node. In this experiment, the function node consists of $\{and, <, >\}$. The terminal nodes are two kinds of symbols; T_1, T_2. The set of T_1 is the name of rules defined in the SpamAssassin. The initial value of T_2 are given by a random number in $[0,1]$. In this paper, there are 300 individuals in GP. The number of agents in each GP is 50 respectively.
For the 3,007 E-mails judged as Spam by SpamAssassin, in which there were 2,913 Spams and 94 Hams misjudged as Spams, ADG can detect all of 94 Ham messages as No-Spam. In this experiment, the extracted rules to judge the Ham message correctly by ADG are shown in Fig.10. However, ADG could not judge correctly four messages of Spam as shown in Fig.11. That is, these four E-mails were misjudged as Ham, even if the correct judgment of messages is Spam. This shows excessive adaptation to the identification of Ham by ADG.

898 554 *** 629 862 448 533 *** 416 *** 696 702 992 703 871 239 *** 977 *** 634 839 *** 876 682 *** *** 878 481 194 470
960 256 *** 373 801 500 *** *** *** 867 438 914 579 *** 851 354 *** *** 188 *** *** *** 268 *** *** 709 733 128 *** 669
*** *** 916 707 823 *** 214 *** 739 *** *** 60 657 *** 967 *** 85 *** *** 744 *** 805 *** *** 4 328 *** 449 *** 272
664 *** *** 284 *** 611 *** *** 316 610 957 *** *** *** *** 850 800 *** 777 *** *** 693 771 *** *** 653 216 *** *** 936
35 807 *** *** *** *** 783 *** 725 *** *** 864 287 *** 352 *** 109 *** 746 *** 182 *** 567 *** *** 564 396 *** 473 376
984 517 426 *** 818 933 644 398 *** 857 825 *** *** *** 896 *** 598 219 948 *** 758 *** *** 973 665 728 329 *** *** 235
824 *** 147 *** 662 *** *** *** *** *** 280 233 *** 663 139 *** 723 727 *** 281 701 736 *** *** 726 982 692 *** 655 906
955 *** 966 *** *** *** 30 915 *** 756 584 *** *** 534 526 195 *** *** *** 884 *** 36 *** *** 940 826 *** 972 912 ***
613 *** *** 522 343 *** 799 *** *** 57 941 *** *** *** 668 *** *** 716 932 *** 229 710 890 *** *** 528 *** *** *** 803
920 698 808 *** 934 678 *** 179 843 *** 493 *** 885 *** 676 169 695 *** *** 22 *** *** *** 552 *** 419 *** *** 913 ***
971 *** 196 *** 372 *** 874 814 903 460 *** 713 985 *** *** *** *** 845 904 722 995 *** *** 495 853 *** 257 922 894 765
367 155 604 *** 829 582 269 *** *** *** 127 946 *** *** 353 *** 164 827 836 180 *** *** 510 *** *** *** *** 660 *** 199
*** 348 *** 622 305 *** 635 535 632 *** *** 532 596 394 *** *** 770 38 536 627 *** 91 828 306 935 556 *** 279 *** 859
784 *** *** *** 371 793 *** *** *** 378 981 *** *** *** *** *** 26 586 *** 775 *** 292 797 986 356 124 *** *** 842 983
462 844 *** 435 *** 541 *** 587 *** *** 418 954 *** 304 *** 901 *** *** *** *** *** 241 763 804 *** 523 768 841 *** 834
905 991 *** 860 *** *** 964 *** 790 379 *** *** 791 *** 406 631 *** 461 *** 558 445 *** 98 640 *** 463 881 *** *** 464
781 *** *** 362 *** 135 *** 151 *** *** 987 *** 961 838 *** *** *** 251 *** 403 762 *** *** 551 962 *** 650 *** 670 ***
*** 326 *** 875 *** 730 546 926 *** *** *** 947 *** 965 751 121 *** 580 603 *** 953 296 *** *** *** *** *** *** *** 720
996 963 *** 994 *** *** *** 200 887 997 *** 911 958 721 208 *** 683 *** 110 *** 364 *** 699 *** *** 734 *** *** *** 260
413 *** *** *** 767 *** 869 *** 959 *** *** 576 *** 649 330 *** *** *** 779 918 *** 891 968 *** *** 224 14 *** *** 74
877 786 *** 848 592 931 *** 502 *** 944 491 409 433 685 *** 927 900 *** *** *** 939 *** 152 *** *** *** 846 *** 163 810
*** *** *** *** 530 612 *** 350 989 852 *** *** 559 *** 925 *** *** 780 119 *** *** 956 469 909 902 *** *** *** *** ***
206 *** 226 *** 341 626 *** 484 690 *** 494 976 614 *** 752 978 *** 375 *** 527 813 *** 923 *** *** *** 276 103 993 393
266 *** 774 *** 636 704 *** *** 737 802 249 *** *** *** 880 *** 773 *** *** *** 854 *** *** *** 654 *** 202 322 542 ***
550 *** 748 *** *** 452 *** 943 *** *** *** *** 930 988 12 *** 79 444 571 840 315 *** *** 743 *** 431 *** 616 *** 830
680 *** *** 897 *** *** 585 568 *** 509 519 339 *** *** *** *** 999 492 919 *** 738 *** 970 *** 772 *** *** *** *** ***
*** 159 847 508 112 *** 677 856 *** *** 593 192 630 *** 715 *** *** 171 *** *** *** *** *** 49 *** 223 *** 975 *** 432
893 *** 455 *** 882 485 *** *** *** 889 168 *** *** 849 *** 866 *** *** *** 863 2 *** *** 430 750 *** *** 518 *** 573
86 *** 597 71 320 *** *** 942 872 *** 949 *** 607 *** 754 *** *** 928 *** 974 *** *** 201 *** *** 231 *** *** *** ***
980 *** 248 562 499 *** 671 689 *** 617 594 *** 952 998 656 *** 895 681 *** 938 *** 555 798 *** 865 717 424 177 835 990

Fig. 9 Classification result of E-mails by SOM

```
Rule0 :
( ISO2022JP_BODY < -0.0257 ) and ( CHINANET < 0.7433 )
and ( URLRBLJP99 < 1.4865 ) and ( DYN_ONEGAI < 1.9846 )
and ( RCVD_IN_BL_SPAMCOP_NET < 0.1000 ) and ( URIBL_SC_SURBL < 4.4980 )

Rule1:
( BAYES_99 < 0.5507 ) and ( SHIFT_JIS2 < 0.1101 ) and ( DEETO < 0.0073 )
and ( USENBROAD < 0.0407 ) and ( RAZOR2_CF_RANGE_E4_51_100 < 0.1101 )
and ( UNDISC_RECIPS < 0.0618 ) and ( RIPE_NCC < 0.0997 )
and ( RCVD_IN_SBL < 0.0337 ) and ( URIBL_WS_SURBL < 0.1571 )
and ( FORGED_MUA_OUTLOOK < 0.2978 )
and ( HTML_OBFUSCATE_05_10 < 0.0073 ) and ( FUAN < 0.0147 )
and ( WINDOWS_CHARSET < 1.4665 ) and ( ADDRESS_IN_SUBJECT < 0.0391 )
and ( DREAMX < 1.4954 ) and ( RIMA_TDE_NET < 0.0734 )
and ( FORGED_RCVD_IP < 0.7343 ) and ( URIBL_OB_SURBL < 0.0073 )
```

Fig.10 Extracted rules by ADG

```
MisJudgement1
X-Spam-Flag: YES
X-Spam-Score: 9.215
X-Spam-Level: ********
X-Spam-Status: Yes, score=9.215 tagged_above=2 required=6.31
    tests=[ANATA=0.5, AWL=2.543, BAYES_50=0.001, CLICK_JP=1,
    CONTENT_TYPE_PRESENT=-0.1, DATE_IN_PAST_06_12=0.827, FORGED_RCVD_HELO=0.135,
    FROM_EXCESS_BASE64=1.309, FUAN=0.2, HAISHINTEISHI=0.3, HTML_MESSAGE=1,
    ISO2022JP_BODY=-0.1, MULTIPART_ALTERNATIVE=0.1, MURYOU=0.2, ONEGAI=0.2,
    PLING_QUERY=0.1, QENCPTR1=0.2, SUBJECT_ENCODED_TWICE=0.1, SUPPORT=0.1,
    TOOLONGSTR=0.5, X_MAILER_PRESENT=0.1]

MisJudgement2
X-Spam-Flag: YES
X-Spam-Score: 10.754
X-Spam-Level: *********
X-Spam-Status: Yes, score=10.754 tagged_above=2 required=6.31
    test=[AISHOU=0.2, ANATA=0.5, AWL=-3.046, BAYES_99=7.5, BUSINESS=0.2,
    CONTENT_TYPE_PRESENT=-0.1, HOTERU=0.5, ISO2022JP_BODY=-0.1,
    ISO2022JP_CHARSET=-0.1, MATTERU=0.3, MURYOU=0.2, OBSCURED_EMAIL=0.1,
    QENCPTR1=0.2, SUBJECT_ENCODED_TWICE=0.1, UNPARSEABLERELAY99=3.5,
    UNPARSEABLE_RELAY=0.5, UWAKI=0.3]
```

Fig.11 Misjudged Spam as Ham

5.2 Extraction of rules by using some trained SOMs

The precise classification of E-mails will require huge amounts of E-mails and the form of Spam will change into various ways every day. However, our method has a limitation in the amount of E-mails to extract classification knowledge once, because SOM cannot afford to train huge amounts of E-mails and ADG will be faced to a difficult judgment to extract a more important rule. Therefore, we improve our proposed method to use two or more SOMs simultaneously and then the portions of classification results by SOM are integrated into the terminals for rule extraction by ADG. Fig.12 shows the overview of our improved method by using some SOM modules and ADG. After scanning E-mails by SpamAssassin, the matching score of rules in the header of E-mail is recorded. As SpamAssassin has various rulesets described in the section 2 to detect a Spam, our improved method use two or more SOM modules which are trained for each rulesset. We can see some divided regions on the maps in each trained SOM. The part of rule falled into each divided region is given as the term such as $\{body0, body1, \cdots, header, \ldots, uri\}$ and then ADG maximizes the fitness f defined as follows:

$$f = -\frac{miss\_\text{target}\_data}{N_{\text{target}}} - \alpha \frac{misrecognition}{N_{non\_\text{target}}} - \beta \frac{\sum_{N_{non\_\text{target}}} fault\_agent}{misrecognition \times N_{agent}} - \delta V_w$$

In this paper, there are 300 individuals in GP. The number of agents in each GP is 100 respectively. The parameters of fitness functions are as follows: $\alpha = 1.0$, $\beta = 0.0001$, $\delta = 0.01$. ADG was trained for the random selected 500 cases.

We performed two kinds of experiments. In our experiment, the aim is to find E-mails with $score < 20$ from Spams. In the other experiments, the targets are E-mails with $score \geq 20$.

As the experimental result, we found that the correct ratio of classification capability by ADG was 84.0% if the total score of SpamAssassin is less than 20, otherwise about 86.6%. The sample of extracted rules are listed in the Table 1 and Table 2.

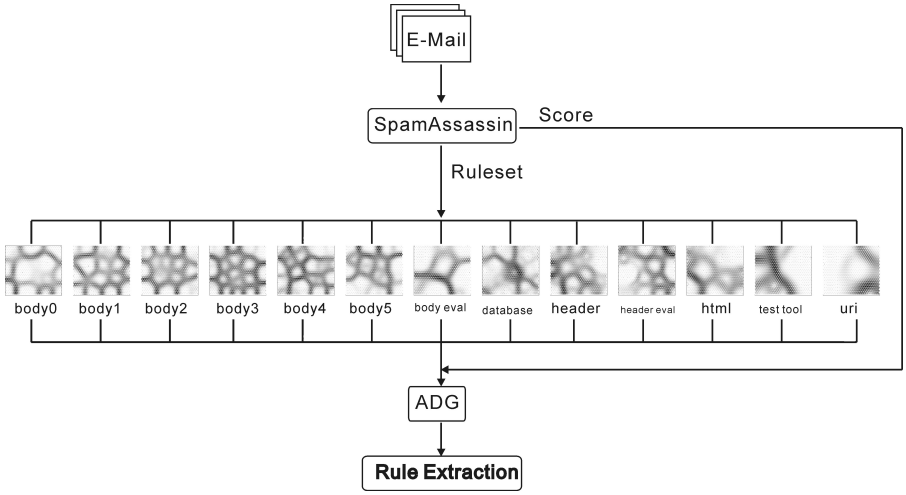


Fig.11 An overview of improved method by using some SOM modules

ID	#Agent	Rule
1	90	(and (eq header 4)(eq database 3))
2	5	(eq testtool 5)
3	4	(and (eq bodyeval 4)(and (eq header 8)(eq database 3)))
4	1	(and (eq header 3)(eq database 3))

Table 1. The judgement rule in case of $score < 20$

ID	#Agent	Rule
1	48	(and (eq testtool 2)(eq database 4))
2	14	(and (eq database 1)(eq testtool 2))
3	13	(and (eq database 2)(eq testtool 2))
4	7	(eq database 5)
5	3	(eq bodyeval 3)
6	2	(eq header 9)
7	2	(eq header 1)
8	2	(eq headereval 4)
9	2	(and (and (eq headereval 1)(eq uri 1))(eq bodyeval 1))
10	1	(eq html 4)
11	1	(eq html 2)
12	1	(eq header 10)
13	1	(eq headereval 3)
14	1	(eq header 2)
15	1	(eq body3 12)
16	1	(eq testtool 1)

Table 2. The judgement rule in case of $score \geq 20$

6. Conclusion

In this paper, we propose a classification method for Spam E-mail based on the results of SpamAssassin. This method can learn patterns of Ham and Spam E-mails. First, SOM can classify many E-mails into the some categories. In this phase, we can see the characters of current received Spam E-mails. Second, ADG can extract the correct judgment rules of Hams misjudged as Spams. However, there are a few cases of Spam misjudged as Ham. In this experiment, ADG makes an over fitting to the characters of Hams. We have met the problems according to the limitation of classification capability by SOM and explosive search in GP using many nodes as shown in T_1 Therefore, we improve the proposed method to classify the Spam E-mails by using some SOM modules and to extract some rules by ADG according to the classification results.

Moreover, these methods give the input patterns using the judgment rules in SpamAssassin and their agreement value. Even if one rule is fire and the other rules are not fire, and the agreement value is high, the result of judgment will be Spam. The SpamAssassin is very useful open source software, the learning tool is equipped in itself. But we may meet the misjudgment. The rule must be redefined by each user according to user's work style and so on. In order to improve such problems, the natural language processing and the mining of important word from E-mails will be required. We will develop the hybrid classification and rule extraction method using the learning of neural networks and ADG.

7. References

The Apache SpamAssassin Project. SpamAssassin http://spamassassin.apache.org/

Kohonen, T.(1995). *Self-Organizing Maps*, Springer Series in Information Sciences, Vol. 30, Springer, Berlin, Heidelberg, New York

Hara, A. & Nagao, T. (1999). Emergence of cooperative behavior using ADG; Automatically Defined Groups, Proc. of the 1999 Genetic and Evolutionary Computation Conf., pp.1039-1046

Hara, A.; Ichimura, T.; Takahama, T. & Isomichi, Y. (2004). Discovery of Cluster Structure and The Clustering Rules from Medical Database Using ADG; Automatically Defined Groups, In *Knowledge-Based Intelligent Systems for Healthcare*, Ichimura, T. & Yoshida, K. (Ed.), Advanced Information and Knowledge Processing, pp.51-86

Hara, A.; Ichimura, T.; & Yoshida K. (2005). Discovering multiple diagnostic rules from coronary heart disease database using automatically defined groups, *Journal of Intelligent Manufacturing*, Vol.16, No.6, pp.645-661

Hara, A.; Kurosawa, Y.; Ichimura, T. (2008). Automatically Defined Groups for Knowledge Acquisition from Computer Logs and Its Extension for Adaptive Agent Size, In *Current Trends in Intelligent Systems and Computer Engineering*, Springer, pp.15-32

Frederick P. Brooks, Jr. The Postfix Home Page, http://www.postfix.org/

Amavisd-new. The amavisd-new Web site, http://www.ijs.si/software/amavisd/

Clam AntiVirus. The Clam AntiVirus Web site, http://www.clamav.net/

Ichimura, T.; Oeda, S.; Suka, M. & Yoshida, K. (2004). A learning method of immune multi-agent neural networks, *Neural Computing and Applications Journal*, Vol.14, pp.132-148

Ichimura, T.; Oeda, S.; Suka, M.;Hara, A.; Mackin, K.J. & Yoshida, K. (2005). Knowledge Discovery and Data Mining in Medicine, In *Advanced Techniques in Knowledge Discovery and Data Mining(Advanced Information and Knowledge Processing)*, Nikhil, P.; Jain, L.C. Ed., Springer, pp.177-210

Yamaguchi, T.; Ichimura, T.; & Mackin, K.J. (2008). Adaptive Tree Structured Clustering Method using Self-Organizing Map, *Proc. of Joint 4th International Conference on Soft Computing and Intelligent Systems and 9th International Symposium on advanced Intelligent Systems (SCIS & ISIS 2008)*, pp.1407-1411

Yamaguchi, T.; Ichimura, T.; & Mackin, K.J. (2008). Analysis using adaptive tree structured clustering method for medical data of patients with coronary heart disease, *Proc. of 4th International Workshop on Computational Intelligence and Applications 2008 (IWCIA 2008)*, pp.139-144

Ichimura, T.; Kurosawa, Y. & Hara, A. (2007). A classification Method for Spam E-mail by Self-Orgaizing Map and Automatically Defined Groups, *Proc. of the 2007 IEEE Intl. Conf. on System, Man and Cybernetics (SMC2007)*, pp.2044-2049

25

Applying an SOM Neural Network to Increase the Lifetime of Battery-Operated Wireless Sensor Networks

Mario Cordina and Carl James Debono
University of Malta
Malta

1. Introduction

Wireless sensor networks have garnered significant attention in recent years. According to (The Mobile Internet, 2004), more than half a billion nodes will be shipped for wireless sensor applications in 2010, for an end user market worth at least $7 billion.

Wireless sensor networks are one of the first real-world examples of pervasive computing, the notion that small, smart, computing and cheap sensing devices will eventually permeate the environment (Bulusu & Jha, 2005). The combination of distributed sensing, low power processors and wireless communication enables such technology to be used in a wide array of applications such as habitat monitoring and environment monitoring, military solutions, such as battlefield surveillance, and commercial applications, such as monitoring material fatigue and managing inventory.

A wireless sensor network consists of hundreds or thousands of low-power, low-cost multi-functioning sensor nodes operating in an unattended environment with a limited supply of energy. The latter is one of the main constraints of each sensor node together with the limited processing power. These limitations, coupled with the deployment of a large number of sensor nodes, pose a number of challenges to the design and management of these networks, requiring energy-awareness at all layers of the networking protocol stack.

The issues related to the physical and link layers are generally common for all sensor applications and therefore research in these areas focused on system-level energy awareness such as dynamic voltage scaling (Heinzelman et al, 2000a), radio communication hardware (Min et al, 2000), low duty-cycle issues (Woo & Culler, 2001), system partitioning (Ye et al, 2002) and energy-aware MAC protocols (Shih et al, 2001). At the network layer, energy efficient route setup protocols are necessary to reliably relay data from the sensor nodes to the sink whilst maximising the lifetime of the network. This chapter will focus on such a solution based on sensor node clustering, whereby the topology is decided through an SOM neural network.

2. Related Work

Since wireless sensor networks often consist of a large number of sensor nodes which have to be networked together, conventional techniques such as the direct transmission protocol have to be avoided because the energy loss incurred can be quite large, depending on the location of the sensor node relative to the base station. Furthermore, using a conventional multi-hop routing protocol such as Minimum Transmission Energy (MTE) (Ettus, 1998; Shepard, 1996) will also result in an equally undesirable effect. This is due to the fact that in MTE, the intermediate nodes are chosen in such a way that the sum of squared distances (and thus the total transmit energy, assuming d^2 power loss) is minimised. Hence for the configuration shown in Figure 1, node A would transmit to node C through node B if and only if:

$$E_{transmit}(d = d_{AB}) + E_{transmit}(d = d_{BC}) < E_{transmit}(d = d_{AC}) \quad (1)$$

or assuming a $\frac{1}{d^2}$ attenuation model,

$$d_{AB}{}^2 + d_{BC}{}^2 < d_{AC}{}^2 \quad (2)$$

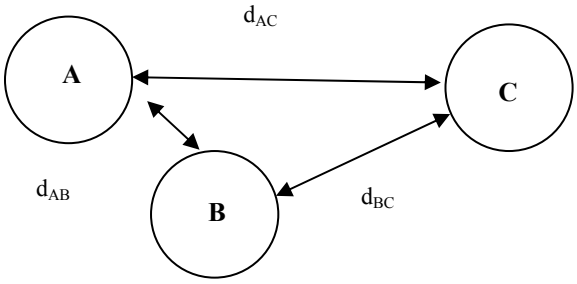


Fig. 1. Minimum Transmission Energy

Thus in MTE, the nodes closest to the base station will rapidly drain their energy resources since they are involved in routing of a large number of messages (on behalf of other nodes) to the base station.

In order to minimise these problems, various routing protocols have been proposed for wireless sensor networks. An in-depth survey of such protocols is presented in (Akkaya & Younis, 2005). Here, Akkaya and Younis classify routing protocols into 3 main categories namely: Location-based protocols, Data-centric protocols and Cluster-based protocols.

Location-based protocols

These protocols utilise the position information to relay data to the desired regions rather than the whole network. An example of location-based protocols is Geographic Adaptive Fidelity (GAF) (Xu et al, 2001).

Data centric protocols

These protocols depend on labelling of the desired data thereby eliminating many redundant transmissions. Two well-known network layer protocols based on data centric

routing are Sensor Protocols for Information via Negotiation (SPIN) (Heinzelman et al, 1999) and directed diffusion (Intanagonwiwat et al, 2000). Although the data centric routing approach provides a reliable and robust solution to wireless sensor networks, there are still some shortcomings associated with it since both SPIN and directed diffusion suffer from the amount of overhead energy spent in activities such as advertising, requesting and gradient setup. Furthermore, the excessive time spent in such activities might not suit applications that require the sensor nodes to respond quickly to an emergency situation.

Cluster-based protocols

In cluster-based routing protocols, nodes are grouped into clusters and each cluster head node collects, processes and forwards the data from all the sensor nodes within its cluster to the base station. The application of clustering-based protocols reduces energy dissipation by selecting optimal cluster heads, thereby reducing transmission distance, and by reducing the amount of information that needs to be transmitted through data aggregation. Research in recent years focused on developing algorithms that select optimal cluster heads in order to reduce the energy dissipation thereby increasing the lifetime of the system.

Cluster-based routing algorithms are generally divided into two categories; centralised and distributed algorithms. One of the most popular cluster-based distributed algorithms is Low-Energy Adaptive Clustering Hierarchy (LEACH) (Heinzelman et al, 2000b). The operation of LEACH is organized in rounds, where each round is composed of a cluster setup phase and a data transmission phase. During the cluster setup phase, nodes organize themselves into clusters with one node serving as the cluster head in each cluster. The selection of a cluster head is decided locally within each node, making LEACH a completely distributed algorithm requiring no global knowledge of the network. To become a cluster head, each node *n* selects a random number between 0 and 1 and if the number is less than the threshold *T(n)* the node is elected as a cluster head for the current round. The threshold *T(n)* is given by:

$$T(n) = \begin{cases} \dfrac{P}{1 - P\left(r \bmod \dfrac{1}{P}\right)} & \text{if } n \in G \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

where, *P* is the cluster head probability, *r* is the number of the current round and *G* is the set of nodes that have not been cluster-heads in the last $1/P$.

During the transmission phase, the self elected cluster heads collect data from nodes within their respective clusters and apply data fusion before forwarding the fused data directly to the base station. At the end of a given round, a new set of nodes become cluster heads for the subsequent round. This cluster head rotation mechanism coupled with data fusion in the cluster head results in LEACH achieving a factor of 7 reduction in energy dissipation compared to direct communication and a factor of 4 - 8 reduction in energy when compared to the minimum transmission energy routing protocol (Akkaya & Younis 2005).

Although LEACH is normally considered as a benchmark protocol, the algorithm used in the selection of cluster heads has a number of disadvantages:

1. Since each node uses probability techniques to decide whether or not to become a cluster head, there might be cases were two cluster heads are in close proximity of each other increasing the overall energy dissipation in the network.
2. The number of cluster head nodes generated is not fixed and so in some rounds it may be more or less than the optimal value.
3. The nodes selected can be located in areas which are not densely populated resulting in further energy expenditure to transmit data to that cluster head.
4. Each node has to calculate the threshold and generate the random number in each round, consuming CPU cycles.
5. LEACH stochastic cluster head selection is prone to lead to energy imbalance in the network. This results in nodes dying at an early stage reducing the network lifetime significantly.

In order to tackle some of the disadvantages found in LEACH, the authors of (Heinzelman et al, 2002) proposed a centralized version called LEACH-C. Unlike LEACH, where nodes self-configure into clusters, LEACH-C uses the base station for cluster formation. During the cluster setup phase, each node sends information about its current location, using a GPS receiver, and energy level to the base station. The energy level is used by the base station to ensure that only the nodes with relatively high energy are participating in the cluster head selection. Using these high energy nodes, the base station uses the simulated annealing algorithm (Maruta & Ishibuchi, 1994) to find k optimal cluster heads in order to minimize the amount of energy dissipation required by the other nodes to transmit their data to the cluster head.

Although the data transmission phase of LEACH-C is identical to that of LEACH, results indicate a 40% increase in data per unit energy over LEACH. The authors of (Heinzelman et al, 2002) cite two key reasons for this improvement:

1. The base station utilises its global knowledge of the network to produce better clusters which require less energy for data transmission.
2. The number of cluster heads in each round of LEACH-C equals a predetermined optimal value, whereas for LEACH the number of cluster heads varies from round to round due to the lack of global coordination among nodes.

Nevertheless, this improvement comes at the expense of having all the sensors equipped with GPS receivers which apart from increasing the cost of each sensor, introduces further energy dissipation.

3. Algorithm Design

3.1 Requirements and Assumptions

In order to find a good compromise between the centralised and distributed approaches, we will now present an SOM neural network-based clustering algorithm. In this solution, the sensor nodes collect network topological information, through localised interactions, which is then transferred to the base station through a low energy cost network initialisation phase. As opposed to other centralised algorithms such as LEACH-C, the initialisation phase allows the base station to have updated network topology awareness without the use of GPS-assisted sensors and without high energy cost procedures.

The design of the algorithm needs to achieve the following targets:

- Ease of deployment: Sensor networks may contain hundreds of nodes and may need to be deployed in remote and dangerous environments. Thus nodes must be small, cheap and able to self-configure with limited global control to setup or maintain the network.
- System lifetime: Sensor nodes must have low energy consumption in order to allow the network to operate for as long a period as possible. Moreover, the energy balancing mechanisms help in extending the system's lifetime.
- Latency: Data from sensor networks is typically time-sensitive, so it is important to receive the data in a timely manner. Long delays due to processing or communication may be unacceptable.

In order to reach these goals, the algorithm mechanisms described below and illustrated in Figure 2 can be applied.

System Lifetime: To improve system lifetime, mechanisms need to be developed to minimise energy dissipation and improve energy balancing between the nodes. Such mechanisms include: Data reduction algorithms based on classic adaptive filters, cluster head separation, neural assisted cluster head election, cluster head rotation and load balancing cost functions.

Latency: Network latency increases as number of nodes between source and destination increases. To reduce latency, the single hop mode of operation can be used for intra-cluster communication, thereby allowing the base station to receive data in a timely manner. Furthermore, the use of data reduction algorithms discussed later on, allow the base station to predict sensor measurements itself thus eliminating any latency issues.

Ease of deployment: The use of a homogenous network structure whereby all the sensor nodes are identical in terms of battery energy and hardware complexity allows for a fast and easy deployment. Moreover, the algorithm can be designed in such a way that nodes self-configure with limited global control to setup or maintain the network.

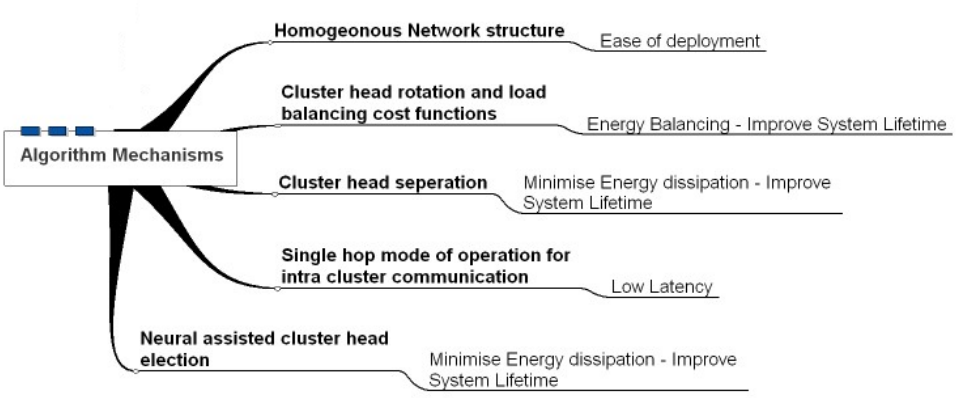


Fig. 2. Algorithm Mechanisms

Prior to a detailed description of the algorithm, it is worth noting the following general assumptions:

- N sensors are uniformly dispersed within a square field of size M x M.
- All sensors and base station are stationary after deployment.

- The intra-cluster communication is based on the single-hop mode of operation.
- Communication is symmetric and a sensor can compute the approximate distance based on the received signal strength if the transmission power is given. If the transmission power is not known, the received signal strength is used as a distance metric.
- All sensors are of equal significance having equal energy and hardware complexity – homogenous network.
- All sensors are location-unaware.

3.2 Radio Propagation Model

The transmission of data from the sensors uses air as a medium and therefore we need a model to estimate the signal levels reaching the receivers. The standard radio model used in wireless sensor networks is based on the fact that the propagation of electromagnetic waves can be modelled as a decaying power law function of the distance between the transmitter and the receiver. Furthermore, if there is no direct line-of-sight path between the transmitter and the receiver, the electromagnetic wave will undergo reflection, diffraction and scattering off objects in the environment. This will result in electromagnetic waves arriving at the receiver from different paths and at different times, leading to multi-path fading, which again can be roughly modelled as a power law function of the distance between the transmitter and the receiver (Heinzelman et al, 2000b).

The radio model commonly adopted to model propagation in wireless sensor networks uses both the free space model and the multi-path fading model, depending on the distance between the transmitter and the receiver. If the distance between the transmitter and a receiver is less than a certain crossover distance $d_{crossover}$, the Friss free space model is used whereas if the distance is greater than $d_{crossover}$, the two-ray ground propagation model is used (Heinzelman et al, 2000b). The crossover distance is defined as follows (Heinzelman et al, 2000b):

$$d_{crossover} = \frac{4\pi\sqrt{L}h_T h_R}{\lambda} \tag{4}$$

where,

$L \geq 1$ is the system loss factor and is not related to propagation,

h_T is the height of the transmitting antenna above ground level in metres,

h_R is the height of the receiving antenna above ground level in metres,

λ is the wavelength of the carrier signal in metres.

When the transmitter and receiver have direct line-of-sight communication, which will only occur if the transmitter and receiver are close to each other (i.e. $d \leq d_{crossover}$), the transmit power is attenuated according to the Friss free space equation as follows (Rappaport, 1996):

$$P_r(d) = \frac{P_t G_t G_r \lambda^2}{(4\pi d)^2 L} \tag{5}$$

where,

$P_r(d)$ is the received power given a transmitter-receiver separation of d in metres,

P_t is the transmit power,

G_t is the gain of the transmitting antenna,

G_r is the gain of the receiving antenna,

d is the distance between the transmitter and the receiver in metres.

If the distance is greater than $d_{crossover}$, the transmit power is attenuated according to the two-ray ground propagation equation as follows (Heinzelman et al, 2000b):

$$P_r(d) = \frac{P_t G_t G_r h_t^2 h_r^2}{d^4} \tag{6}$$

In this case, the received signal comes from both the direct path and a ground-reflection path leading to a higher transmit power attenuation rate proportional to d^4.

Although the standard radio model is widely used in modelling and simulation of wireless sensor networks, it is rather simplistic. As the credibility of high level protocol simulation results depends on the accuracy of the physical layer model (Kotz et al, 2004), more accurate radio models are required. A review of studies carried out in this area reveals a general lack of near-ground channel measurements, as the vast majority of the studies place antennas at heights which are greater than one metre. However, recently studies were carried out in this area and radio models based on field measurements were specifically developed for wireless sensor network. In (Fanimokun & Frolik, 2003), Fanimokun and Frolik present specific models for near ground wireless sensor networks operating in the 915 MHz ISM band based on the single slope log normal shadowing model. Similarly in (Molina-Garcia-Pardo et al, 2005), the authors present models for an operational frequency of 868 MHz based on the two slope log normal models. Although these models are more accurate than the standard radio model discussed earlier, they are specific to the environment in which the measurements were taken and therefore they cannot be reliably applied to different scenarios.

3.3 Energy Model

The standard energy consumption model used in wireless sensor networks is that developed by (Heinzelman et al, 2000b). Heinzelmann adopts a simple energy model where the transmitter dissipates energy to run the radio electronics and the power amplifier, whilst the receiver dissipates energy to run the electronics as shown in Figure 3.

Furthermore, this energy model makes use of the standard propagation model whereby the power attenuation depends on the distance between the transmitter and receiver. Power control is used to invert this loss by adjusting the power amplifier to ensure that a certain signal power level reaches the receiver.

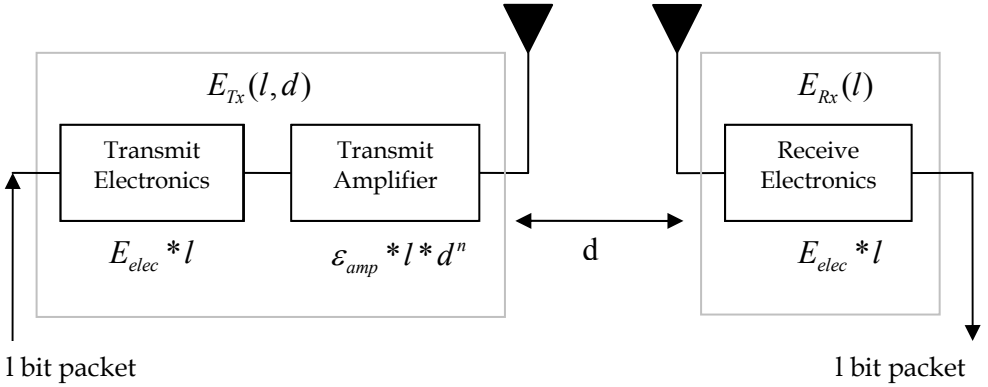


Fig. 3. Radio energy dissipation model (Heinzelman et al, 2000b)

Thus, to transmit an *l*-bit message a distance *d*, the radio expends:

$$E_{TX}(l,d) = E_{TX-Elec}(l) + E_{TX-amp}(l,d) \tag{7}$$

This is equivalent to:

$$E_{TX}(l,d) = \begin{cases} lE_{Elec}(l) + l\varepsilon_{fs}d^2 & ; d < d_{crossover} \\ lE_{Elec}(l) + l\varepsilon_{mp}d^4 & ; d \geq d_{crossover} \end{cases} \tag{8}$$

And to receive this message, the radio expends:

$$E_{Rx}(l) = E_{Rx-elec}(l) \tag{9}$$

$$E_{Rx}(l) = lE_{Elec}(l) \tag{10}$$

The electronics energy E_{Elec} depends on factors such as digital coding, modulation and filtering of the signal before it is sent to the transmit amplifier. The typical value used for E_{Elec} is 50 nJ/bit (Heinzelman et al, 2000b). The parameters ε_{fs} and ε_{mp} depend on the receiver sensitivity and noise figure, as the transmit power needs to be adjusted so that the power at the receiver is above a certain threshold $P_{Threshold}$ (Heinzelman et al, 2000b).

3.4 Algorithm Details

The general operation of the clustering algorithm is divided into three phases as shown in Figure 4.

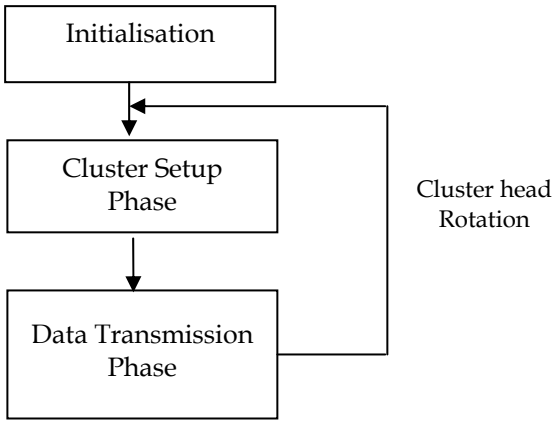


Fig. 4. General operation of the Algorithm

Similar to LEACH, the operation of the algorithm is divided into rounds. Each round begins with a cluster setup phase were the clusters are organised, followed by a data transmission phase during which data is transferred from the nodes to the cluster head which aggregates all the data received together with its own data and forwards it to the base station.

Initialisation Phase

During the initialisation phase, the base station calculates important network parameters and collects topological information from the nodes through localised interaction. This phase needs to consume very little energy and therefore it must be designed to use small control packets and minimise the number of transmissions. The operation of the initialisation phase is as follows:

1. The base station calculates the optimal number of cluster heads K_{opt} and the minimum cluster head separation distance R_{sep} .
2. The base station broadcasts a "Network Initialisation" message to all nodes in the network at a fixed power $P_{Broadcast}$ which includes the minimum separation distance R_{sep} .
3. All the nodes in the network receive this broadcast and use it as a 'beacon' signal to calculate their distance from the base station d_{TOBS} by considering the received signal strength as a distance metric.
4. Each node then broadcasts a "Neighbourhood Search" message including their node ID within a range R_{sep} by setting the transmit power to P_{Search}.
5. All the nodes in the network listen for the "Neighbourhood Search" messages and record all the received messages, storing the Node ID and received signal strength.
6. Following the "Neighbourhood Search" procedure, each node uses the recorded "Neighbourhood Search" messages to calculate its node centrality and concentration as in (Gupta et al, 2005). These parameters together with the list of neighbouring nodes and the distance metric to each node are then transmitted to the base station.

Optimum number of cluster heads

The optimal number of cluster heads, K_{opt}, can be analytically determined using the computation and communication energy models as in LEACH (Heinzelman et al, 2000b). The authors of (Heinzelman et al, 2000b), considered the case where the cluster head aggregates all the received data into a single packet by performing data aggregation. However, this is not realistic and the case where the cluster head compresses all the received data into a packet of size ρl is considered.

Assume that there are N nodes distributed uniformly in an M x M region. If there are k clusters, each cluster has an average of N/k nodes (one cluster head and $(N/k) - 1$ non-cluster head nodes). Now, let $\rho \in [0,1]$ be the compression factor, l be the length of data packets transmitted from the node to the cluster head, P be the number of non-cluster head nodes in a cluster and L_M be the length of the compressed data packet transmitted from the cluster head to the base station.

Then,

$$L_M = \left(P - \left(P - 1\right)\rho\right)l \tag{11}$$

Since the base station is typically far from the nodes, it is assumed that the energy dissipation of the cluster head follows the multi-path model (d^4 power loss). Thus, the energy dissipated in the cluster head during a single frame is given by:

$$E_{CH} = \left(\frac{N}{k} - 1\right)lE_{elec} + L_M\left(E_{elec} + \varepsilon_{mp} {d_{TOBS}}^4\right) \tag{12}$$

where E_{elec} is the electronics energy taken to be 50nJ/bit (Heinzelman et al, 2000b), ε_{mp} depends on the receiver sensitivity and noise figure, and d_{TOBS} is the distance from the cluster head node to the base station.

Each non-cluster head node transmits its data to the cluster head once during a frame. As the distance to the cluster head is small, it is assumed that the energy dissipation of the non-cluster head node follows the Friss free space model (d^2 power loss). Thus the energy dissipated in each non-cluster head node is:

$$E_{non-CH} = lE_{elec} + l\varepsilon_{fs}{d_{TOCH}}^2 \tag{13}$$

where ε_{fs} depends on the receiver sensitivity and noise figure and d_{TOCH} is the distance from the non-cluster head node to the cluster head. As shown in (Heinzelman et al, 2000b),

$$E[{d_{TOCH}}^2] = \frac{M^2}{2\pi k} \tag{14}$$

Therefore, the energy dissipation of the non-cluster heads is given by:

$$E_{non-CH} = lE_{elec} + l\varepsilon_{fs}\frac{M^2}{2\pi k} \tag{15}$$

Furthermore, the total energy dissipated in a cluster during a frame is:

$$E_{cluster} = E_{CH} + \left(\frac{N}{k} - 1\right)E_{non-CH} \approx E_{CH} + \frac{N}{k}E_{non-CH} \tag{16}$$

Hence the total energy dissipated in a frame is given by:

$$E_{total} = kE_{cluster} = k\left[E_{CH} + \frac{N}{k}E_{non-CH}\right] \tag{17}$$

Substituting (12) and (13) in (17) yields:

$$E_{total} = nl\left[2E_{elec} + \varepsilon_{fs}\frac{M^2}{2\pi k}\right] - klE_{elec} + kL_M\left[E_{elec} + \varepsilon_{mp}{d_{TOBS}}^4\right] \tag{18}$$

The optimal number of cluster heads is found by setting the derivative of E_{total} with respect to k to zero, yielding:

$$k_{opt} = \sqrt{\frac{nl\varepsilon_{fs}M^2}{2\pi\left[\rho lE_{elec} - 2lE_{elec} - l\varepsilon_{mp}{d_{TOBS}}^4 + \rho l\varepsilon_{mp}{d_{TOBS}}^4 + \left[E_{elec} + \varepsilon_{mp}{d_{TOBS}}^4\right]\rho l\right]}} \tag{19}$$

Minimum cluster head separation distance

In order to reduce the energy dissipation of the sensor nodes, cluster heads are spread across the whole field by considering a minimum cluster head separation distance, R_{sep}. This ensures that no two cluster heads are in close proximity of each other, thereby increasing network lifetime.

Consider K_{opt} cluster heads distributed uniformly in an M x M region. The minimum cluster head separation distance R_{sep} is given by:

$$R_{sep} = \sqrt{\frac{M^2}{\pi k_{opt}}} \tag{20}$$

Determining $P_{Broadcast}$ and P_{Search}

With reference to equations (6), (7), (8) and assuming the following radio parameters: $f = 914MHz$, $\lambda = 0.328m$, $G_T = G_R = 1$, $L = 1$, $h_T = h_R = 1.5m$ (Heinzelman et al, 2000b), the crossover distance is given by:

$$d_{crossover} = \frac{4\pi\sqrt{L}h_T h_R}{\lambda} = 86.2m \tag{21}$$

Thus,

$$P_R = \begin{cases} \left[\frac{G_T G_R \lambda^2}{(4\pi)^2}\right]\frac{P_T}{d^2} = 6.82 * 10^{-4}\frac{P_T}{d^2} & \text{if } d < d_{crossover} \\ G_T G_R h_T^2 h_R^2 \frac{P_T}{d^4} = 5.1 * \frac{P_T}{d^4} & otherwise \end{cases} \tag{22}$$

Assuming that the receiver's thermal noise floor is 99dBm, the receiver noise figure is 17dB and assuming that an SNR of at least 30dB is required to receive the signal with no errors, then the minimum received signal level for successful reception is given by (Heinzelman et al, 2000b): $P_R \geq$ -99 + 17 + 30 = -52$dBm \rightarrow 6.3\mu W$.
Substituting P_R in (22) yields,

$$P_T = \begin{cases} 9.24 * 10^{-3} d^2 & \text{if } d < d_{crossover} \\ 1.24 * 10^{-6} d^4 & otherwise \end{cases} \tag{23}$$

In general, to determine P_{Search} we consider $d < d_{crossover}$ whereas to determine $P_{Broadcast}$ we consider $d \geq d_{crossover}$.

Cluster Setup Phase

During the cluster setup phase, the base station has to elect K_{opt} cluster heads. This decision is based on four parameters, namely:

- *Node Residual Energy:* The elected node must have a high residual energy as the cluster heads are involved in high energy consumption operations. These nodes have to aggregate the data and transmit larger packets to the base station.
- *Centrality:* The node centrality defines a value which classifies the node importance based on how central the node is to the cluster. The elected node needs to have a low centrality value indicating that the node is central in relation to the surrounding nodes thereby reducing the energy dissipation between the nodes and the cluster head.
- *Concentration:* This defines the number of nodes present in the vicinity. The elected node must have a high node concentration thereby ensuring that cluster heads are elected where they are mostly required.
- *Cluster head frequency:* In order to avoid electing the same node repeatedly, the base station has to keep a record of the number of times each node was elected as cluster head.

The election of cluster heads is carried out using an SOM neural network (Kohnen, 1995). The neural network uses the above mentioned parameters to partition the nodes according to their cluster head quality. From this we define high quality cluster heads as those sensor nodes that present high residual energy, low centrality, high concentration and low cluster head frequency.

A 4 input - 16 output neuron SOM topology is adequate to solve this cluster head selection problem (Cordina & Debono, 2008). The network is used to partition data into 16 categories

depending on the cluster head quality as shown in Figure 5. The weights $W_{i,j}$ are initially set such that neuron Y_1 corresponds to high quality nodes whereas neuron Y_{16} corresponds to low quality nodes. These weights are then fine tuned using the standard SOM training methodology based on 10,000 normalised test vectors.

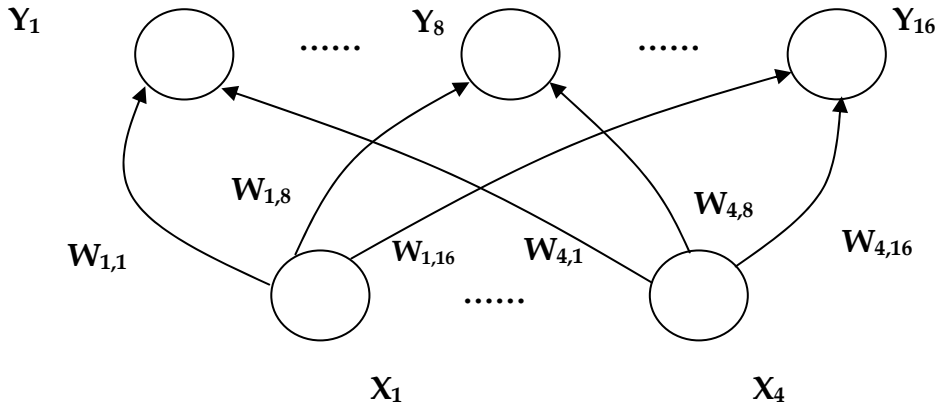


Fig. 5. Self Organising Map structure

The general operation of the cluster head election mechanism is shown in Figure 6.

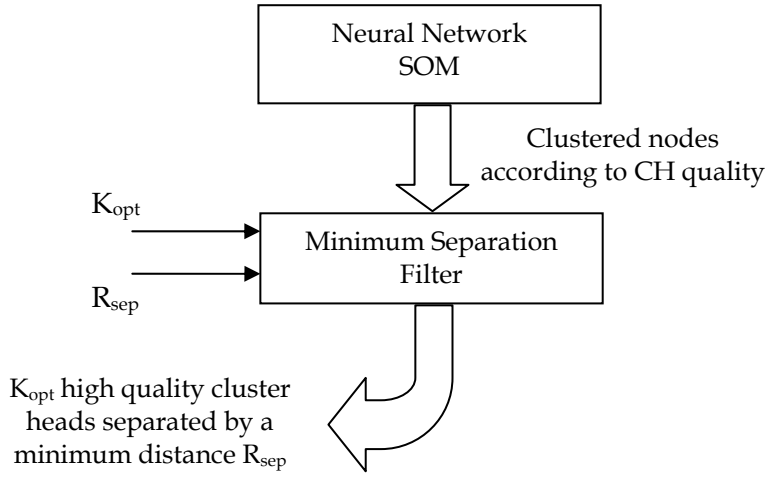


Fig. 6. Cluster head election mechanism

The output from the neural network is a list of clustered nodes sorted according to their cluster head quality. This list is then applied to a minimum separation filter which ensures that a minimum separation distance is guaranteed between the elected cluster heads. This

yields K_{opt} high quality cluster heads which are separated by the defined minimum distance R_{sep}.

The operation of the cluster setup phase is as follows:

1. During the initialisation phase, the base station collects all the information sent from the nodes and compiles a table as shown in Table 1. Note that the initial energy of each node is predefined and the cluster head frequency, which defines the number of times the node is elected as cluster head, is initially set to zero.

Node ID	Centrality	Concentration	Energy	CH Frequency	Distance to BS	Neighbours
1	0.34	1	0.5	0	80.5	54
2	0.532	3	0.5	0	75.2	5, 10, 14
3	0.784	2	0.5	0	97.6	65, 88
…	…	…	...	…	…	…
…	…	…	...	…	…	…

Table 1. Information collected by the base station

2. The Centrality, Concentration, Energy and CH frequency for all the nodes are extracted from Table 1, normalised (Doherty et al, 2007) and applied to the neural network to cluster the nodes according to their cluster head quality. Following the clustering process, the filtering algorithm is applied to the output of the neural network.
3. Using Table 1, the base station computes the cost function parameters $d_{g\_\min}$ and $d_{g\_\max}$ on the chosen CHs as follows

$$d_{g\_\min} = \min(d_{TOBS})$$
$$d_{g\_\max} = \max(d_{TOBS})$$

4. The base station informs the selected cluster head nodes for the current round and also includes the cost function parameters.
5. The selected cluster heads broadcast an "Advert CH" message to all the nodes in the network at a fixed power $P_{Broadcast}$ including their Node ID and cost function parameters.
6. Nodes in the network receive the "Advert CH" messages and compute the cost for each cluster head. Various cost functions can be used, two of which are shown here:

Standard cost function based on received signal strength – In most of the algorithms, nodes choose the cluster heads based on the received signal strength in order to lower their energy consumption. However as discussed in (Ye et al, 2005), this approach may lead cluster head nodes to exhaust their energy rapidly during the data transmission phase as cluster head nodes which are further away from the base station have a higher energy consumption compared to those being in the vicinity of the base station.

Advanced cost function based on a weighted distance-energy metric – This cost function presented in (Cordina & Debono, 2009) considers both the distance metric and the energy of the cluster head node relative to that of the node. Thus, the cost of a cluster head node which is far

away from the base station is offset according to the energy of the cluster head node relative to that of the node. The cost function is given by:

$$Cost(j,i) = w_1 * f\left(d\left(P_j, CH_i\right)\right) + \left(1 - w_1\right) * g\left(d\left(CH_i, BS\right)\right) - \left(1 - w_2\right) * \frac{E_{CH}}{E_{Node}}$$
$$f = \frac{d\left(P_j, CH_i\right)}{d_{f\_\max}}; g = \frac{d\left(CH_i, BS\right) - d_{g\_\min}}{d_{g\_\max} - d_{g\_\min}} \quad (24)$$

where, E_{CH} and E_{Node} are the cluster head and node residual energy respectively, w_1 and w_2 are the distance and energy cost function weights respectively, $d_{g\_\min}$ and $d_{g\_\max}$ are cost function distance parameters described earlier and $d_{f\_\max} = Ex[\max\{d(P_j, CH_i)\}]$ is a normalising factor for the distance between the node P_j and the cluster head CH_i.

7. Each node selects the cluster head which has the minimum cost and sends a "Join Request" message to the selected cluster head node including Node ID, cost and status of the exclusive flag. The latter is used by the node to inform the cluster head that it has received only one "Advert CH" message, this indicates that the node is on the fringe of the network. The exclusive flag is then used to prioritise between the nodes.
8. Cluster heads receive the "Join Request" messages and notify the nodes of their acceptance.

The nodes whose energy level falls below the Critical Energy Level whilst in the cluster setup phase send a "Node dead" message directly to the base station. Similarly, elected cluster head nodes whose energy level falls below the Critical Energy Level during the cluster setup phase, send a "Cluster Head dead" message directly to the base station. This allows the base station to update the status of the nodes in the network during the cluster setup phase. If the base station detects that a cluster head is 'dead', it marks its status 'dead' and broadcasts a re-cluster message to the whole network indicating that a re-cluster is necessary. On receiving a 'Re-clustering' message, all the nodes disassociate themselves from their cluster head and the cluster setup phase is re-started.

Data Transmission Phase and cluster head rotation

During the data transmission phase, the nodes perform measurements and append status and residual energy information to each packet before transmitting it to their cluster head. The cluster heads receive this data, append their status and residual energy information, compress it and forward the resulting packet to the base station. This allows the base station to update the energy level and status of all the nodes in the network. Whenever the base station detects that a cluster head is 'dead', it broadcasts the re-cluster message.

As there is a cost in terms of time and energy associated with the cluster setup phase, the data transmission phase should be long when compared to this phase to reduce the effect of the overhead incurred during cluster formation on the overall performance. On the other hand, as the energy at each node is limited, running the data transmission phase for too long drains the energy of the cluster head node and curtails communication between the non-cluster head nodes that still have energy and the base station. For this reason, a cluster head rotation mechanism is employed to balance energy between the nodes. In this algorithm, a cluster head rotation is triggered when the cluster head residual energy in the current round falls below the residual energy in the previous round by some percentage hysteresis. On a

cluster setup phase re-start, the base station has an updated status of the network and recalculates the optimal number of cluster heads K_{opt} to ensure optimality throughout the lifetime of the network. All these energy overheads related to the initialisation phase, cluster setup phase, associated re-clustering procedures and the appending of status/residual energy to the transmitted packets must be considered in the simulations.

3.5 Results

Prior to fully testing the algorithm, the two SOM neural network parameters, namely the learning rate and sigma, need to be optimised. Simulations are initially carried out on a fixed network in order to determine these optimal parameters. The clustering performance of the neural network is assessed by computing the overall cluster quality index as defined in (He et al, 2003). With reference to Figure 7, the optimal SOM neural network parameters are: learning rate=0.1 and sigma=0.2.

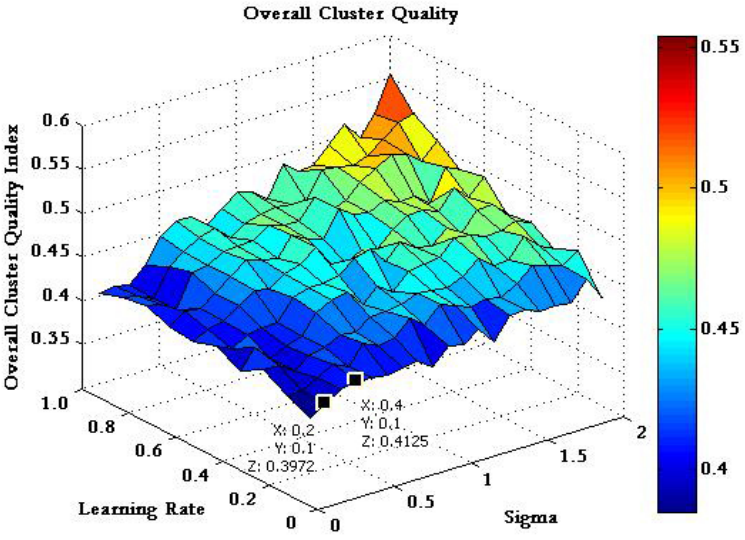


Fig. 7. Overall cluster quality index

Using these optimal SOM neural network parameters, the full algorithm can be tested in a wireless sensor network simulator, in this example the simulator is implemented in MATLAB®. Simulations were performed on 100 nodes uniformly dispersed in a 100m by 100m field with the base station located at coordinates (150m, 50m). The energy parameters used in this simulation are tabulated in Table 2.

Initial Battery energy	0.5 Joules
Energy model parameter, ε_{fs}	$1*10^{-11}$
Energy model parameter, ε_{mp}	$1.3*10^{-15}$
Electronics Energy, E_{elec}	50nJ/bit
Data packet length	4000bits
Control packet length	200 bits

Table 2. Assumed energy parameters

The performance of the algorithm can be assessed along three metrics. These are (1) *First Node Dies (FND)* which defines the time taken for the first node to die, (2) *Half Nodes Alive (HNA)* which defines when half the network has died, and (3) *Transmitted Packets (TP)* which gives the total number of transmitted packets by the sensor nodes at FND. To evaluate the validity and performance of the algorithm, simulations are carried out over 50 randomly generated 100 node networks. The improvement over LEACH in terms of mean and standard deviation values of the performance metrics is shown in Table 3.

Metric	**Standard Cost Function**		**Advanced Cost Function**	
	Mean	Std. Deviation	Mean	Std. Deviation
FND	51.1%	4.8%	65.2%	4.6%
HNA	34.%	0.9%	32.8%	0.8%
TP	27.6%	4%	36.1%	3.3%

Table 3. Performance Metrics

Algorithm	**Improvement over LEACH**
EEPSC (Zahmati et al, 2007)	45%
LEACH-C (Heinzelman, 2002)	62.5%
LEACH-Deterministic (Handy et al, 2003)	30%
This SOM-based Algorithm	65.2%

Table 4. Performance Improvement of various algorithms

These results show that the algorithm is capable of creating better clusters, increasing the energy balancing between the nodes whilst lowering the rate of energy dissipation. This in turn results in a significant improvement in the FND and HNA, leading to an increase in the number of packets received by the base station. Furthermore, the use of a cost function based on a weighted distance-energy metric (with w_1 and w_2 set to 0.9) further optimises the energy dissipation and energy distribution between the nodes leading to an additional improvement in the network lifetime, with an increase of up to 65.2% in FND over LEACH

as shown in Table 3, and a reduction in the FND to HNA transition time. A graphical representation comparing LEACH with the proposed algorithm is illustrated in Figure 8 while the superiority of this algorithm, based on an SOM neural network, with respect to other solutions is shown in Table 4.

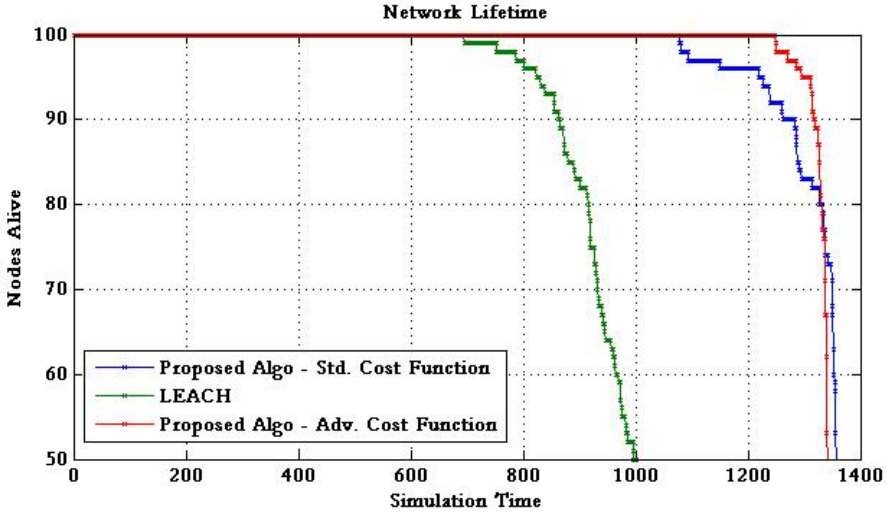


Fig. 8. Network Lifetime

4. Conclusion

We have presented a cluster-based routing algorithm developed using an SOM neural network architecture together with a cost function based on a weighted distance-energy metric. Simulation results show the efficacy of the algorithm as it manages to improve the system lifetime by up to 65.2% when compared to LEACH. Furthermore, by optimising the distribution of the energy between the nodes, a reduced FND to HNA transition time is achieved thereby improving the network quality. This solution makes wireless sensor networks more attractive as they can remain operational for a longer period of time.

5. References

The Mobile Internet (2004), *Wireless Sensor networking: $7 billion market by 2010*. [Online]. Available: http://findarticles.com/p/articles/mi_m0NZB/ is_4_6/ ai_n6054921

Akkaya K.; Younis M. (2005). A survey of Routing Protocols for Wireless Sensor Networks *Elsevier Ad Hoc Network Journal,* Vol. 3, pp. 325-349, 2005

Bulusu, N.; Jha, S. (2005). *Wireless sensor networks – A System Perspective*, Artech House, Boston, 2005

Cordina, M.; Debono, C.J. (2008). Increasing Wireless Sensor Netowrk Lifetime through the Application of SOM Neural Networks, *Proceedings of the 2008 3rd International Symposium on Communications, Control and Signal Processing,* March 2008

Cordina, M.; Debono, C.J. (2009). Maximizing the Lifetime of Wireless Sensor Netowrks through Intelligent Clustering and Data Reduction Techniques, *Proceedings of the 2009 IEEE Wireless Communications and Networking Conference,* April 2009

Doherty K.; Adams R.; Davey N. (2007). Unsupervised Learning with Normalised Data and non-Euclidean Norms, *Applied Soft Computing*, vol.7, no.1, pp.203-210, January, 2007

Ettus M. (1998). System Capacity, Latency and Power Consumption in Multihop-Routed SS-CDMA Wireless Networks, *Proceedings of IEEE Radio and Wireless Configuration*, August 1998

Fanimokun A; Frolik J. (2003). Effects of Natural Propagation Environments on Wireless Sensor Network Coverage Area, *Proceedings of the 35th IEEE Southeastern Symposium on System Theory*, Morgantown, West Virginia, USA, March 2003

Handy M.; Haase M. & Timmermann D. (2003). Low Energy Adaptive Clustering Hierarchy with Deterministic Cluster-Head Selection, *Proceedings of the Fourth IEEE Conference on Mobile and Wireless Communications Networks*, Stockholm, September, 2003

He J.;Tan A.; Tan C. & Sung S. (2003). On Quantitative Evaluation of Clustering Systems, *Information Retrieval and Clustering*, W. Wu, H. Xiong, and S. Shekhar, Eds., Boston, MA, 2003

Heinzelman W.; Kulik J. & Balakrishnan H. (1999). Adaptive Protocols for Information Dissemination in Wireless Sensor Networks, *Proceedings of the Fifth Annual ACM/IEEE International Conference on Mobile Computing and Networking*, 1999

Heinzelman W., et al. (2000a). Energy-Scalable Algorithms and Protocols for Wireless Sensor Networks, *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP '00),* Istanbul, Turkey, June 2000

Heinzelman, W.; Chandrakasan A. & Balakrishnan H. (2000b). Energy-efficient Communication Protocol for Wireless Sensor Networks, *Proceedings of the Hawaii International Conference System Sciences*, Hawaii, January 2000

Heinzelman W.; Chandrakasan A. & Balakrishnan H. (2002). An Application Specific Protocol Architecture for Wireless Micro-sensor Networks, *IEEE Trans. Wireless Communications*, vol. 1, no. 4, pp. 660-670, October 2002

Intanagonwiwat C.; Govindan R. & Estrin D. (2000). Directed Diffusion: A Scalable and Robust Communication Paradigm for Sensor Networks, *Proceedings of ACM MobiCom*, August 2000

Kohnen, T. (1995). *Self Organising Maps*, Springer-Verlag, Berlin, 1995

Kotz D.; Newport C.; Gray B.; Liu J.; Yuan Y. & Elliott C. (2004). Experimental Evaluation of Wireless Simulation Assumptions, *Proceedings of the 7th ACM international symposium on Modelling, analysis and simulation of wireless and mobile systems (MSWiM'04)*, Venice, Italy, October 2004

Maruta T.; Ishibuchi H. (1994). Performance Evaluation of Genetic Algorithms for Flowshop, *Proceedings of the 1st IEEE Conference on Evolutionary computation*, June 1994

Min R., et al. (2000). An Architecture for a Power Aware Distributed Micro-sensor Node, *Proceedings of the IEEE Workshop on signal processing systems (SIPS'00),* October 2000

Molina-Garcia-Pardo J.; Martinez-Sala A.; Bueno-Delgado M.; Egea-Lopez E.; Juan-Llacer L. & Garcıa-Haro J. (2005). Channel Model at 868 MHz for Wireless Sensor Networks in Outdoor Scenarios, *International Workshop on Wireless Ad-hoc Networks,* London, UK, May 2005

Rappaport, T. (1996). *Wireless Communications: Principles & Practice*, Prentice Hall, New Jersey, 1996

Shepard T. (1996). A Channel Access Scheme for Large Dense Packet Radio Networks, *Proceedings of ACM SIGCOMM 1996*, Stanford University, August 1996

Shih E., et al. (2001). Physical Layer Driven Protocol and Algorithm Design for Energy-efficient Wireless Sensor Networks, *Proceedings of the 7th Annual ACM/IEEE International Conference on Mobile Computing and Networking,* Rome, Italy, July 2001

Woo A.; Culler D. (2001). A Transmission Control Scheme for Media Access in Sensor Networks, *Proceedings of the 7th Annual ACM/IEEE International Conference on Mobile Computing and Networking,* Rome, Italy, July 2001

Xu Y.; Heidemann J. & Estrin D. (2001). Geography-informed Energy Conservation for Ad Hoc Routing, *Proceedings of the ACM/IEEE International Conference on Mobile Computing and Networking*, Rome, Italy, July, 2001

Ye M.; Li C.; Chen G. & Wu J. (2005). EECS: An Energy Efficient Clustering Scheme in Wireless Sensor Networks, *Proceedings of IEEE International Performance Computing and Communications Conference (IPCCC)*, 2005

Ye W.; Heidemann J. & Estrin D. (2002). An Energy-Efficient MAC Protocol for Wireless Sensor Networks, *Proceedings of IEEE Infocom 2002,* New York, June 2002