Turk J Elec Engin, VOL.12, NO.1 2004, © TÜBİTAK

Real-Time Classification Algorithm for Recognition of Machine Operating Modes by Use of Self-Organizing Maps

Gancho VACHKOV
Department of Reliability-based Information Systems Engineering,
Faculty of Engineering, Kagawa University,
Hayashi-cho 2217-20, Takamatsu-Shi, Kagawa-ken 761-0396-JAPAN
e-mail: vachkov@eng.kagawa-u.ac.jp
Yuhiko KIYOTA
Faculty of Engineering, Kagawa University,
Hayashi-cho 2217-20, Takamatsu-Shi, Kagawa-ken 761-0396-JAPAN
e-mail: kiyota@eng.kagawa-u.ac.jp
Koji KOMATSU
Faculty of Engineering, Kagawa University,
Hayashi-cho 2217-20, Takamatsu-Shi, Kagawa-ken 761-0396-JAPAN
e-mail: komatsu@eng.kagawa-u.ac.jp
Satoshi FUJII
Hydraulic Excavator Development Center, Shin Caterpillar Mitsubishi LTD,
Akashi-Shi, Hyogo-ken 674-8686-JAPAN

Abstract

In this paper a new algorithm for classification and real-time recognition of different a-priorily assumed operating modes for construction machines is proposed. This algorithm utilizes the effectiveness of the Self-Organizing Maps (SOM) for creating the so called Separation Models, that are able to distinguish each operating mode separately. After training, these models are used in a real-time procedure, which calculates at each sampling time the minimal Euclidean distances from the current data point to a certain node of each SOM. Then the separation model (represented by a respective SOM) that has the least minimal distance to this data point defines the class of the current operating mode.

Simulation results and extensive analysis, based on experimental data from a hydraulic excavator have shown that the proposed algorithm outperforms the standard one-model approach. It is faster in the terms of computation time for training and leads to a higher percentage of true recognitions.

Key Words: *Classification, Self-Organizing Maps, Real-Time Recognition, Operating Modes, Separation Models*

1. Introduction

Many industrial systems and especially the construction machines often work in different and changing operating modes rather than in a steady-state regime. The frequent and sometimes periodical changes of

the operating modes are more demanding for the machine from a viewpoint of reliability. The fast changing operating modes also require a proper evaluation of the current technical condition of the machine that is further used in the decision for the next maintenance stop. In order to analyze the current performance of the machine as well as the beginning of the trend of its deterioration, all the operating modes should be properly recognized for the purpose of the further performance analysis of the machine.

It is supposed that an appropriate sensory system is available and attached to the machine for measuring, saving and possibly transmitting the different parameter readings during the real-time operation of the machine. Then the problem of the operating modes recognition becomes a *real-time classification problem*, which is basically more difficult than the classical off-line classification.

In this paper a new algorithm for classification and real-time recognition of different (a-priorily assumed) operating modes for construction machines is proposed. This algorithm utilizes the effectiveness of the self-organizing maps (SOM) for construction of the specially introduced *Separation Models*. These models are trained in off-line to distinguish each operating mode separately, based on the previously accumulated experimental data.

Then, in the proposed real-time computation procedure, at each sampling time the minimal Euclidean distance from the current measured data point to a certain node of each SOM is calculated. Then the separation model (and its respective SOM) that has produced the least minimal distance defines the class of the current operating mode.

The rest of the paper is organized as follows. The complexity of the real practical problem for recognition of operating modes is explained in the following Section 2. In Section 3 a reference of the existing methods for classification with their merits and demerits is given. Section 4 contains the outline of the proposed real-time classification method. Section 5 describes the structure and the off-line learning algorithm for the self-organizing maps that are further used as separation models. Section 6 gives detailed explanation of the proposed real-time classification procedure. Some experimental results of real-time classification, based on collected real data are presented in Section 7. The final Section 8 concludes the main results in this paper.

2. The Practical Problem

The hydraulic excavator is a complex machine powered by a turbo-diesel engine and using a complicated hydraulic system for different movements during its normal operation. It usually works in a dynamical sequence of several repetitive *operating modes.* The following several operating modes are typical for the normal operation of the excavator, when it does not change its position on the ground:

Mode 1. Loading the bucket with the raw material (sand, stones etc);

Mode 2. Moving the load to a nearby truck;

Mode 3. Unloading the bucket material into the truck;

Mode 4. Return to the initial position for the next loading with the empty bucket.

Mode 5. Movements of the excavator's bucket for leveling the load on the full truck;

Mode 0. Idling (waiting mode with empty bucket): no movements.

All these modes are quite different from a point of view of load and request for engine power. They have also different impact to the long-term process of a gradually deterioration of the total excavator performance. When the previously defined performance criterion for the excavator is no more satisfied (for example the fuel consumption is going gradually above the required level), then the machine has to be stopped for

maintenance. Since it is a very costly and time consuming procedure, the correct distinction in long-term of all the operating modes for the concrete excavator is able to provide us with valuable information that can be further used to make a proper decision about the time for the necessary maintenance and repair of the machine.

Several parameters are considered to be important in this analysis and are measured in real-time mode by respective sensors, as follows: $P1$- Engine Speed [rpm]; $P2$ – Engine Boost Pressure; $P3$- Engine Oil Pressure; $P4$– Fuel Consumption; $P5$ – Left Hydraulic Pump Pressure and $P6$ – Right Hydraulic Pump Pressure.

Then the statement of this practical problem is to distinguish (classify) the current operating mode of the excavator by using an appropriate automatic procedure based on the real-time measurements from the sensors.

The complexity of this recognition problem is illustrated in the following Fig.1. It shows experimental data, taken by sampling of two parameters: $P1$ and $P2$ of a hydraulic excavator. The respective operating modes are shown as bold step-wise lines with 4 different steps that follow periodically 8 times in the sequence: $1, 2, 3, 4$.

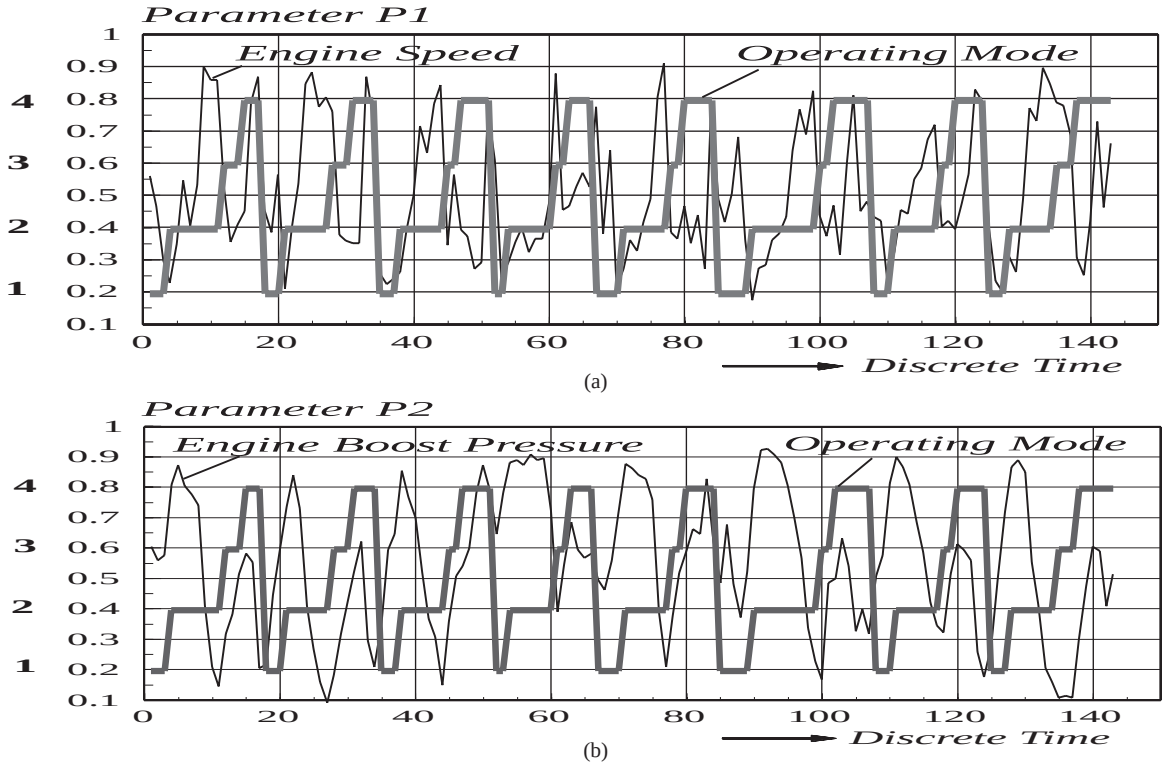

Figure 1. Example of Real-time Measured Data for two Parameters and the Respective Sequence of Four Operating Modes (bold step-wise line).

It is obvious that the proper recognition of the operating modes cannot be done by a simple observation of the dynamic behavior of the parameters. Further look at the two-dimensional plots $P1 - P2$ of the data for all 4operating modes is given in Fig. 2. It shows a wide overlapping area of the observed parameters $P1$ and $P2$ for all the modes that makes the recognition problem difficult and as a result may lead to an ambiguous solution.

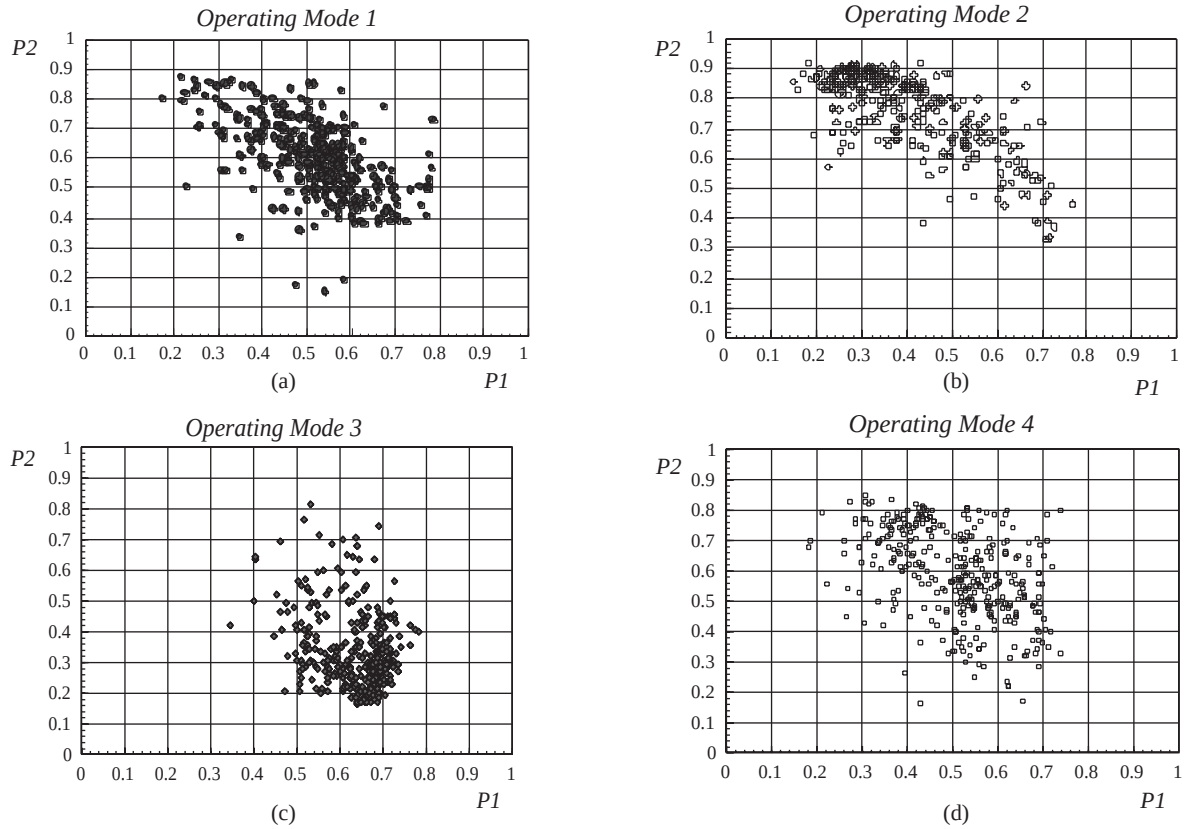

Figure 2. Two-Dimensional Plots of the Parameters $P1$ and $P2$ for All Operating Modes.

3. The Theoretical Problem and Existing Classification Solutions

The practical problem explained above can be viewed as a typical classification problem from a theoretical viewpoint. In the classification problem usually we assume that a total number of DP n-dimensional data points is available: $d(h) = [d_1(h), ..., d_i(h), ..., d_n(h)],\ h = 1, 2, ..., DP$ where $d_i(h),\ i = 1, 2, ..., n$ represents the i-th measured parameter (the i-th feature) of the data point $d(h)$. All DP data points are considered to belong to one of the Z classes (groups): $C_1, ..., C_j, ..., C_z$. In the crisp classification case, each data point belongs to only one class, while in the fuzzy classification one data point belongs basically to all the classes to some degree that is specified by a membership grade.

The classification problem is to build an appropriate model based on some logic, rules, algorithm or heuristics that is able to assign the correct class to each data point. The obtained recognition model is further referred to as *classifier* of this problem.

Figure 3 illustrates the standard approach to the solution of the classification problem, namely the *one-model* approach. Here only one (*global*) *classifier* is previously trained to recognize all possible different classes.

The group of the clustering techniques such as the most popular C-means and fuzzy C-means algorithms [1] seem to be not appropriate tools for real-time classification, because they are used basically in off-line mode in order to distribute the already collected data into a preliminary given number of classes. In addition, it may happen that the data from the same operating mode of the excavator is located in two or even more regions in the space. Then the clustering algorithm will separate such data in different clusters (classes), which does not correspond to the reality.

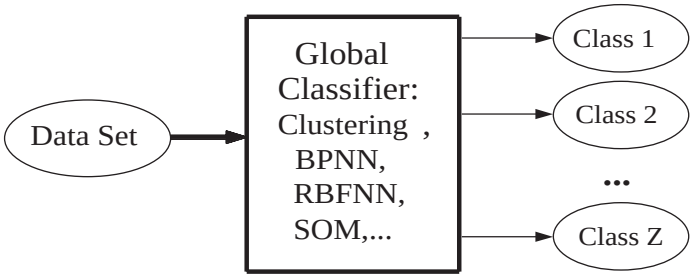

Figure 3. The Standard *One-Model* Approach to Data Classification.

It is clear that the classification model (the classifier) has to be learned in order to capture the peculiarities of each operating mode. Different learning algorithms (training procedures) are used in order to obtain such a classifier.

In the popular "*supervised learning*" procedure a predefined set of $TD \leq DP$ *training* data points with a known "answer" (the actual class) is used for *off-line* training, where at each iteration the classifier gradually updates its parameters.

Frequently used classifiers (recognition models) for solving the standard classification problem are the neural networks, such as the Back-Propagation Neural Networks (BPNN) or the Radial-Basis Function Neural Networks (RBFNN) described in [2,3]. They have been successfully used in different applications [4,5]. However BPNN and RBFNN are basically black-box models that serve as universal data approximators. Therefore they do not support any visualization and additional explanation of the classification results. In such cases the rule-based neural networks [4], including RBFNN provide better explanation of the obtained results than BPNN because of their rule-based structure.

The self-organizing feature maps (*SOM*), first proposed by Kohonen [6-9,11] belong to the group of the network models that use unsupervised learning [10]. In other words the self-organizing maps are able to organize the set of all input patterns into classes independently, i.e., without "teacher" (answer). An important point in the structure of *SOM* is that its neurons have also a spatial property, i.e., they are located on a discrete lattice. Therefore the neurons in the trained *SOM* represent (visualize) the initial high-dimensional input pattern onto the two-dimensional neurons space.

If preliminary information about the classes of the input data is available, then the SOM can be also trained in a supervised way, by *labeling* (coloring) of the neurons after the initial unsupervised training of the *SOM* is completed. The *SOM* so obtained can be further used as a (global) classifier for the new input data that will be classified to its nearest (most plausible) class. All these properties make *SOM* a very convenient tool for solving different practical problems of classification and data visualization, as shown in [10, 12, 13, 14, 15]. The "Neural-Gas Networks" (*NGN*), as presented in [16-20] are further applications of *SOM* that are specially designed as "topology representing networks". The *NGN* are usually learned in an *unsupervised* way [17,18,19] even if both of *supervised* learning and *unsupervised* learning can be applied [20]. The most important feature of the *NGN* is their ability to gradually "grow" in number of neurons (cells) [17,20]. From the practical problem stated in Section 2, it follows that a good classification of the operating modes could be done if the shape of the "data clouds" such as those shown in Fig. 2 could be remembered by the classifier. Therefore both the *SOM* and the *NGN* seem to be appropriate models to be used as a classifier.

In this paper the self-organizing maps (*SOM*) are further chosen as an appropriate tool for creating the specially introduced *separation models* in the proposed classification method.

4. Outline of the Proposed Real-Time Classification Method

In many real cases the available data do not carry enough information for a clear division of the classes. It often happens that some data points are very close or almost coincide with each other (in the feature space) but still belong to different classes. An illustration of such difficult cases for classification was given in Fig. 2 (Section 2.) where overlapping areas are easily noticed in the parameters space for the different operating modes. If such data are further used for supervised learning of a single *global classifier* (possibly a *SOM*) as shown in Fig. 3, they would be not able to provide good information for clear division of all operating modes. Therefore it is expected to be quite difficult and not reliable for one classifier only (one *SOM*) to distinguish clearly all the operating modes.

In the sequel a new algorithm for classification and real-time recognition of different operating modes for construction machines is proposed. The list of all Z operating modes (classes) that has to be classified is assumed initially. For each class C_1, C_2, ... , C_Z , a so called *Separation Model* is created in the form of a single self-organizing map: SOM_1, SOM_2, ... , SOM_Z . Each of these separation models is trained by using data that represent this operating mode only. In such way the separation model SOM_i , $i = 1, 2, ..., Z$ acts as a "local well trained expert" which is able to clearly distinguish that operating mode. At the same time however, this model has not specialized knowledge about the other classes (modes), since it has not been trained to recognize them.

The main idea of the proposed technology for recognition of machine operating modes is shown in the general flowchart in Fig. 4.

There are two main computation steps in the proposed idea as: *Off-line Step* that uses the *off-line* data stream and *Real-time Step* that uses the *real-time* data stream, as shown in Fig. 4.

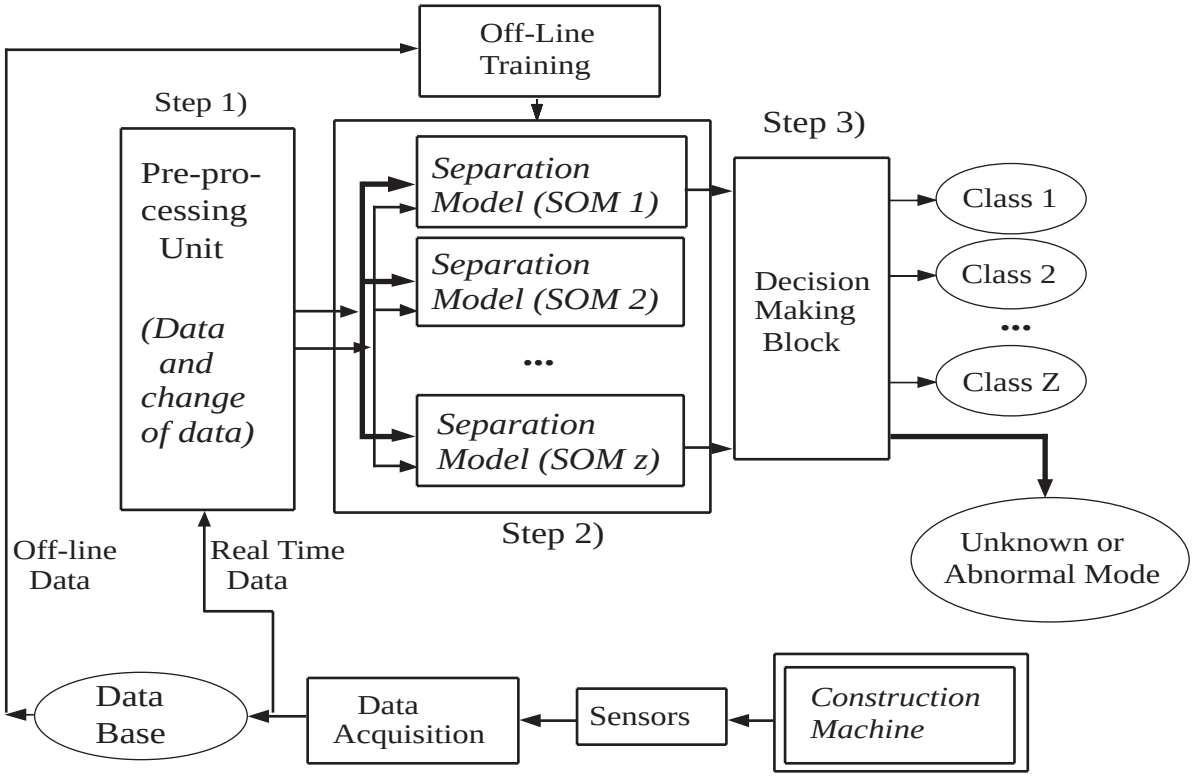

Figure 4. General Flowchart of the Proposed Algorithm for Real-Time Recognition of Machine Operating Modes.

A) *Off-line* Computation step. This is a preliminary computation step at which *off-line* training of the separation models: SOM_1, SOM_2, ... , SOM_Z is performed. For this purpose a large and a reliable experimental data for each operating mode has to be obtained beforehand by experiments. This data is stored in a data base and then used for off-line training of each self-organizing map: SOM_1, SOM_2, ... , SOM_Z.

Basically the training procedure for the separation models that is explained in the next section, is a fast one, since less training data (for one operating mode only) is used, TD_i , $i = 1, 2, ..., Z$, instead of the full set of TD training data. Therefore, generally, a simpler SOM with smaller number of neurons LM_i, $i = 1, 2, ..., Z$ will be needed to define the structure of these models.

As a result of the training, each separation model "captures" the areas of the parameter space that are most typical for this operating mode. These are the areas with the biggest density of data for this mode.

It is obvious that the off-line computation step, as shown in Fig. 4 is the most important and time-consuming computation step that determines the quality of the separation models that are further used for the real-time classification. Therefore a special attention should be paid to the "good" training of the *SOM*. The details are explained in the next Section.

B) *Real-time* Computation Step. This is a step that is repeatedly performed at each data sampling and uses the real-time data stream containing the current sampled data only. Here the computation procedure makes classification and decision for the *Most Plausible* operating mode. In some (more difficult) cases it shows not only one mode, but, rather a list of several "Candidate Operating Modes" that are most similar to the information taken from the current measured data.

5. Structure and the Computational Procedure for Off-line Training of the Self-Organizing Map

As stated above, the off-line computation step aims at creating separation models in the form of self-organizing maps. There are many variations [7,8,9,11] among the general structure and the computational algorithms for training of the self-organizing maps. We keep the general scheme [9,11] with some modifications in the training algorithm as in [12]. Furthermore we assume that a set of TD n-dimensional training data is available, as follows:

$$D = \{d(h)\} = \{ d_1(h), d_2(h), ..., d_n(h) \}, \quad h = 1, 2, ..., TD \tag{1}$$

All of this data form the *data-layer* (lower layer) of the self-organizing map. The *neurons layer* (upper layer) of the map is constructed as a *two-dimensional plane* with evenly distributed neurons in L rows and M columns, as shown in Fig. 5. Thus the total number of all neurons is $LM = L \times M$ and usually $TD >> LM$.

Each neuron in the two-dimensional space is referred to by its *spatial coordinates* $N(i, j)$, $i = 1, 2, ..., L$; $j = 1, 2, ..., M$. Then the *Euclidean* distance $\delta(i, j)$ between a pair of neurons $N(i, j)$ and $N(p, q)$ can be calculated as in [12]:

$$\delta(i, j) = \sqrt{(i - p)^2 + (j - q)^2} \tag{2}$$

At the same time however, the neurons are considered as n-dimensional units since a representative vector $w(i, j)$ is assigned to each of them. The elements of these vectors are the tuning parameters (*weights*) of the neurons, as follows:

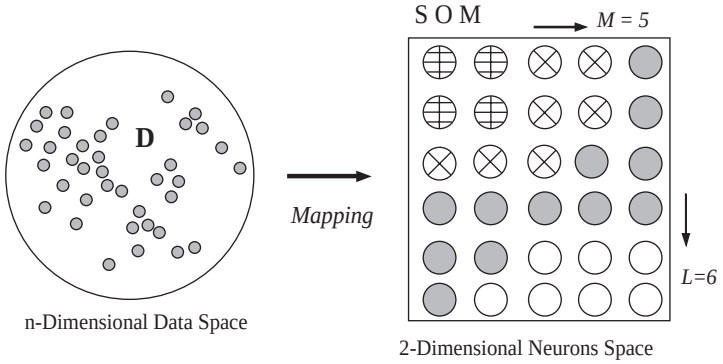

Figure 5. Example of Mapping the $n-$Dimensional *Data Space* onto 2-Dimensional *Neuron Space* with 30 Neurons and 4 Classes.

$$W = \{\, w(i,j) \,\} = \{\, w^1(i,j),\, w^2(i,j), ..., w^n(i,j) \,\}, \\ i = 1, 2, ..., L;\; j = 1, 2, ..., M. \tag{3}$$

Since the data vector $d(h)$ and the neuron weights $w(i,j)$ have the same dimension, the Euclidean distance is used again as a *similarity measure* between the specified data point h and the current neuron $N(i,j)$:

$$\varphi_h(i,j) = \sqrt{\sum_{r=1}^{n} [d^r(h) \,-\, w^r(i,j)\,]^2} \tag{4}$$

Before the start of the *SOM* training, some parameters have to be fixed or initialized in advance. First of all the number of the *training epochs* T should be preliminarily determined and the initial weights (3) for all LM neurons are randomly generated in the n-dimensional space. By fixing the number T of the training epochs we ensure the convergence of the training process for *SOM*, even if sometimes an improper (too low or too big) number T may cause premature stop or over training that leads to an improper classification performance.

During each training epoch $t = 1, 2, ..., T$ all TD data points are presented and the weights of all LM neurons are incrementally adjusted according to the following learning rule

$$w^r_{t+1}(i,j) = w^r_t(i,j) + \Delta w^r_t(i,j),\; r = 1, 2, ..., n. \tag{5}$$

where

$$\Delta w^r_t(i,j) = \alpha_t\, R_t(i,j)\, [d^r(h) - w^r_t(i,j)], \\ i = 1, 2, ..., L; j = 1, 2, ..., M;\; r = 1, 2, ..., n;\; h = 1, 2, ..., TD \tag{6}$$

Here the *learning rate*

$$\alpha_t = a_o(1 \,-\, t/T) \tag{7}$$

is chosen as a monotonically decreasing function (linear in this case), as in [7,9,11,12], in order to guarantee the final convergence of the learning process.

A *Neighborhood Area Function* (*NAF*) $R_t(i,j)$ in (6) is also used during the training of the *SOM* model in order to control the *spatial property* of the self-organized network. It means that the neurons, which are closer to each other, receive more learning increment than the far located neurons. The *NAF* is also called a *kernel* and usually takes the shape of a smooth monotonically decreasing function of the distance from its center. There is a variety of different functional representations of the *kernels* [6-9,12] but from now on we adopt the following *Gaussian*-type function, as in [12]:

$$R_t(i,j) = \exp[-\beta_t \, \delta^2(i,j)], \tag{8}$$

Here the *kernel* changes (decreases) gradually its width in the n-dimensional with the growing number of iterations in the following way:

$$\beta_t = \beta_o \, (t/T)^s \tag{9}$$

The parameter s, $2 \leq s \leq 4$ defines the steepness in the width change of the kernel and β_0 is a predetermined initial parameter ($\beta_0 = 10$ in the further simulations). The term $\delta(i,j)$ in (6) represents the Euclidean distance as in (2) between a neuron $N(i,j)$ and the so called *winner-neuron* $N(p,q)$ for the current presented data point $d(h)$. By definition, the *winner neuron* is the most closely located neuron to the current data point $d(h)$ in the n-dimensional space. The minimum distance:

$$\varphi_h(p,q), = \min_{\substack{1 \leq i \leq L \\ 1 \leq j \leq M}} \{\varphi_h(i,j)\} \tag{10}$$

is calculated by using (4).

Defined in this way, the neighborhood area function in (8) allows bigger learning rate for neurons that are closer to the *winner-neuron* so that such "neighbors" gradually gather closer to the "winner", thus forming a kind of cluster ("clouds" of data). In addition, the size of the "neighborhood" is gradually decreasing in order to ensure the convergence and the generalization ability of the network.

A graphical illustration in Fig. 6 shows how the different tuning parameters influence the learning rate during the training of the *SOM*.

It should be noted that sometimes, after the end of the training procedure with T training iterations, there could be some units of the *SOM*, which haven't been "*winner-neurons*" for any of the *TD* training data points. In other words, these neurons have not contributed to the process of finding the minimal distance to any of the data points. There could be different reasons for such a situation, but most often the reason is the specific (not good) initial distribution of the neurons prior to the training process. Basically such neurons are not useful as representative points of the "clouds of data" because they are far from the data clusters. Therefore we label them as inactive or "*idling neurons*", which are to be deleted from the *SOM* model after the training.

We define two important parameters of the trained *SOM*, that evaluate "how good" is the distribution of the neurons in the data space, namely:

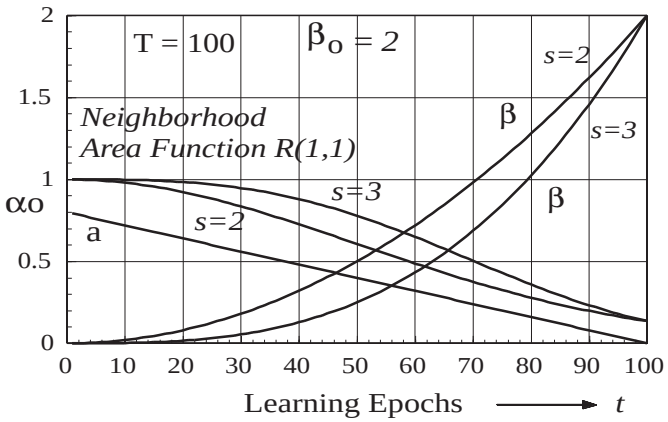

Figure 6. Influence of the Tuning Parameters to the Learning Rate of the Self-Organizing Map.

- the Average Minimum Distance *AVmin* and

- the Standard Deviation *STdev* of the all Minimum Distances $MD(i)$ for the trained *SOM*.

By definition, the Minimum Distance $MD(i)$, $i = 1, 2, ..., TD$ is the minimum of all distances between the $i-$th data point and all the neurons in *SOM*. The $j-$th neuron, for which this distance is found, is further called "winner neuron". In other words:

$$MD(i) = \min_{1 \le j \le LM} \{r(i,j)\},\ i = 1, 2, ..., TD \tag{11}$$

where $r(i,j)$represents the Euclidean distance between the $i-$th data point: $d(i),\ i = 1, 2, ..., TD$ and the $j-$th neuron $n(j)$ in the n-dimensional space. Then the Average Minimum Distance is calculated as:

$$AV_{\min} = \frac{1}{TD} \sum_{i=1}^{TD} MD(i) \tag{12}$$

The following Fig. 7 illustrates graphically an example with minimum distances between 10 data points and 7neurons in a *SOM*. Two neurons, $n(3)$ and $n(7)$ are defined as "Idling Neurons" and eliminated from the *SOM* model.

The Standard Deviation *STdev* for all the minimum distances $MD(i)$, $i = 1, 2, ..., TD$ is another parameter that characterizes the trained *SOM*. It is calculated as

$$STdev = \frac{1}{n-1} \sum_{i=1}^{TD} [\, MD(i) - AV_{\min}]^2 \tag{13}$$

In our off-line training procedure, we propose the idea that the "best" separation model for each operating mode has to be obtained through a kind of *Multi-training Optimization* procedure in which different *SOM*s have to be first trained and then evaluated by *AVmin* and *STdev*. The optimization criterion is aimed at selecting the *SOM* with the least *AVmin* and the preliminarily defined (acceptable) *STdev* between the training data points and the neurons in the *SOM*.

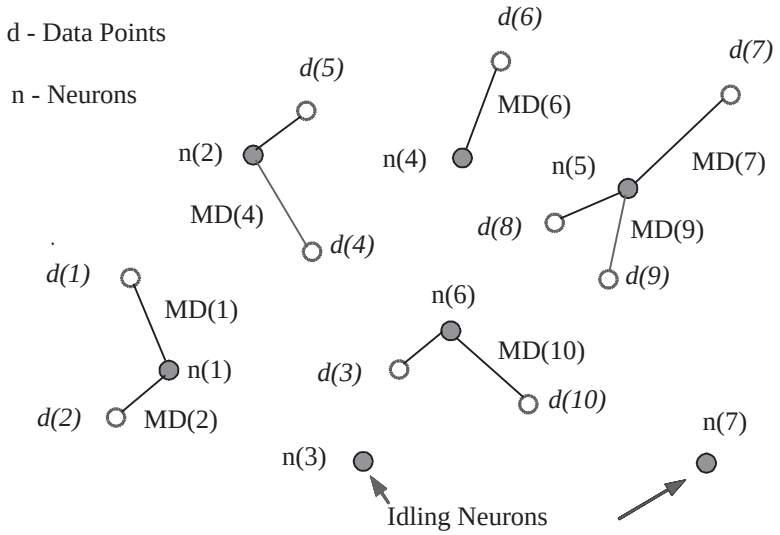

Figure 7. Representation of the Minimum Distances *MD(i)* between 10 Data Points and 7 Neurons on the Self-Organizing Map (*SOM*). Neurons $n(3)$ and $n(7)$ have been Defined as "Idling Neurons".

After the training, the *SOM* for this operating mode consists of neurons that are located closest (in average) to the data points, thus minimizing the average distance to them. In addition, there are more neurons in the areas with bigger concentration of data points than in the areas of sparse data points.

After such arrangement of the neurons in *SOM*, they are further considered as "representative points" for the whole training set of TD data. Therefore the advantage of using *SOM* as a kind of separation model for a specific operating mode can be viewed as follows: The whole data area containing large number of data points characterizing the specific operating mode is approximated by a much smaller number of representative points – neurons. Therefore it is easier to keep them in the memory and makes the real-time classification faster later.

The following Fig. 8 shows graphically the best training results for a *SOM* that is further used as a *Separation Model* for the *Operating Mode 2.* Four different two-dimensional plots are shown with different pairs of parameters. The **p**arameters of the optimized *SOM* are: *AVmin = 0.0628; STdev = 0.0211.* The total number of neurons for the self-organizing map is: $LM = 8 \times 8 = 64$. The number of data points collected for this operating mode is: $TD_2 = 827$.

It is easy to see how the neurons have been placed mainly in the dense areas of the data. Then, in the next (real-time) classification procedure, these neurons will serve as representative points of the whole training data set.

6. The Detailed Steps of the Real-Time Classification Procedure

The proposed real-time classification procedure in Section 4. is performed at each sampling time and consists of three computation steps, as shown in Fig. 4. They are explained in more detail below.

Step B.1) *Preprocessing Step.* Here, after the dynamic process data $d(k)$ are sampled at the current sampling time k, the change-of-the data $\Delta d(k) = d(k) - d(k-1)$ is calculated as a difference between each measured process parameter for the current sampling k and its previous sampling $(k-1)$. In such way, the complete input data pattern is created for the current sampling time.

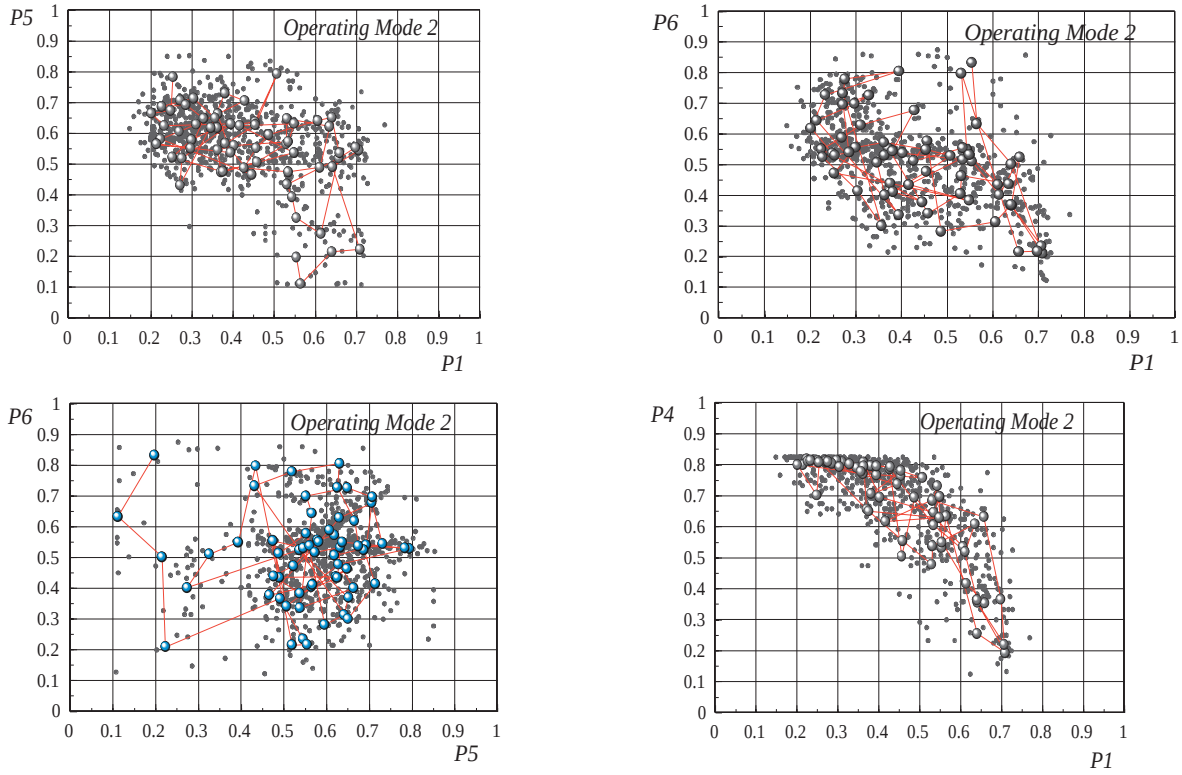

Figure 8. Two-Dimensional Plots of the Original 827 Training Data for *Operating Mode 2* (small dots) and the 64 Neurons Location (bold dots) of the Trained *SOM* with Removed Idling Neurons.

The motivation for adding such additional calculation procedure that defines the differences $\Delta d(k)$, is the fact that we are dealing with dynamical data. It means that only the current measuring does not give sufficient representative information about the dynamics of the process. However, by taking into account the differences, we can "catch" the *tendency* in data trajectories, which could be unique for each operating mode at the current sampling time.

Obviously such additional data parameter $\Delta d(k)$ in the calculation step means that we have to create a *SOM* with twice the original size $d(k)$ and $\Delta d(k)$, which would require more training time. However, the training procedure is done only once in *off-line* mode, which means that training time is not critical for the real-time mode recognition. At the same time, the advantage of this additional preprocessing procedure is in the increased number of "true" classification cases during the real-time recognition.

Step B. 2) Calculation of the "*Current Distance*" $CD(k)$ between the input data pattern at the current sampling time k and a preliminarily specified separation model from the set: SOM_1, SOM_2, ..., SOM_z. This time-variant parameter is calculated as a minimum of the Euclidean distances $r(k,j)$ [7,9,11] between the current data pattern $\{\, d(k);\, \Delta d(k) \,\}$ and a neuron $n(j)$ in this separation model, as follows:

$$CD(k) \;=\; \min_{1 \le j \le LM} \{\, r(k,j) \,\},\; k = 1, 2, ... \tag{14}$$

From a physical point of view, the current distance $CD(k)$ is a kind of measure for the *similarity degree* between the current data pattern $\{\, d(k);\, \Delta d(k) \,\}$ and a given separation model, where the closer distance $CD(k)$ represents a higher similarity degree.

Step B.3) *Decision Making Step.* This is a *post-processing* operation, which is aimed at making the final decision and representing the classification results from the current sampled input pattern $\{ d(k); \Delta d(k) \}$. This step can be performed in different ways, which directly influence the result of classification. We further divide this step into the following processing sub-steps:

3.1. Calculation of the so called "*Relative Distance*" $RD(k)$ between the current input data pattern $\{ d(k); \Delta d(k) \}$ and the winner neuron in a specified separation model by using the following relation:

$$RD(k) \; = \; CD(k) \, / \, AV_{\min} \, , \; k \; = \; 1, 2, ... \tag{15}$$

3.2. If $RD(k) \; > \; 1 \; + \; \alpha$ ($\alpha$is a predetermined *threshold*) for all separation models, this is an indication that the observed input pattern *does not match* any of the preliminarily known operating modes. Then a conclusion (decision) for "Abnormal or Unknown Operating Mode" is displayed. It can be further used as a kind of alarm that initiates the Fault Diagnosis Procedure.

3.3. If, for one or more separation models the calculated relative distance is $RD(k) \; \leq \; 1 \; + \; \alpha$,then a conclusion for *one or more* possible "Candidate Operating Modes" is made. These candidates are further arranged and displayed to the human operator in increasing order of their distance, i.e., in decreasing order of their possibility.

The following Fig. 9 depicts a simulated example of a real-time recognition of four operating modes: $1, 23$ and 4 that have been preliminarily trained as respective separation models. However the real-time data include also the *Idling Operating Mode* 0 which has not been trained (there is no separation model SOM_0 for it). The bold step-wise line shows the actual operating mode at each sampling time, while the thin line shows the recognition of the mode by the proposed method. It is seen that sometimes a discrepancy in the recognition occurs. The typical case is the "misclassification" of the "*Idling Mode*" (0) which is detected as "*Abnormal Mode*" (-1), since there is no training information for such a mode.

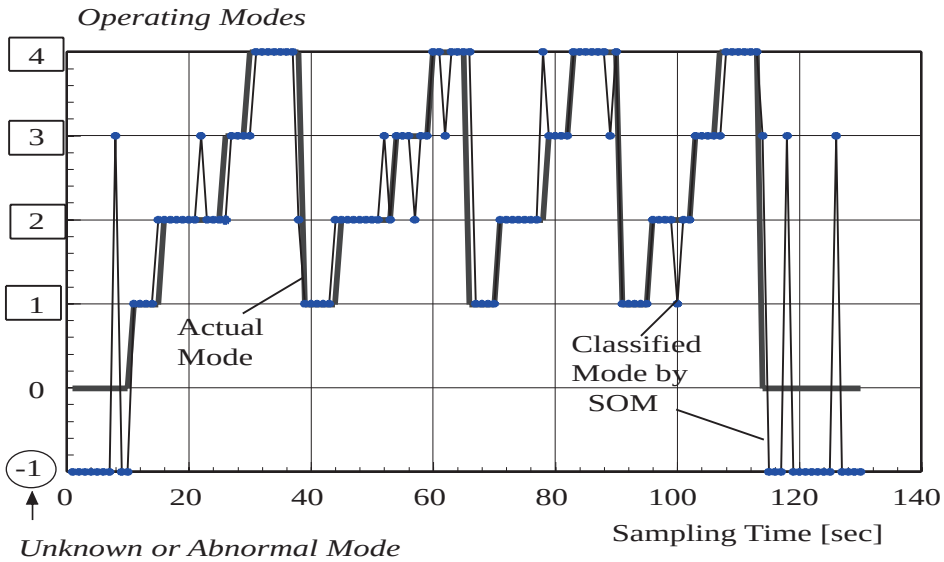

Figure 9. Simulated Example for a Real-time Classification of Four *Trained* and One *Untrained* Operating Modes, Based on the Proposed Classification Algorithm.

7. Experimental Results

The proposed idea and algorithm for real-time classification of the operating modes, based on a fixed number of separation models is investigated on many test examples as well as on the real experimental data from a hydraulic excavator for the operating modes, mentioned in Section 2. Some results from a real-time classification during a time interval of 200 seconds are depicted in Fig. 10.

In this Figure, the trained operating modes are: *Mode 1, 2, 3*, 4 and 5. The idling mode 0 of the excavator has not been trained and as a result, the classification algorithm recognizes it mostly as "unknown" (not seen) or "Abnormal Mode", but sometimes also mixes it with mode 5, because it looks similar by parameters. A recognition level of *87.5%* true classification has been achieved when 4 parameters: P_1, P_2, P_5 and P_6 have been used for creating the separation models SOM_1, SOM_2, ..., SOM_5. Each of them has a total number of neurons $LM = 11 \times 11 = 121$. The experiments with the full number of 6 parameters: P_1, P_2, ..., P_6 have produced slightly better recognition level (*93.8%*) due to increase in the information needed for the recognition.

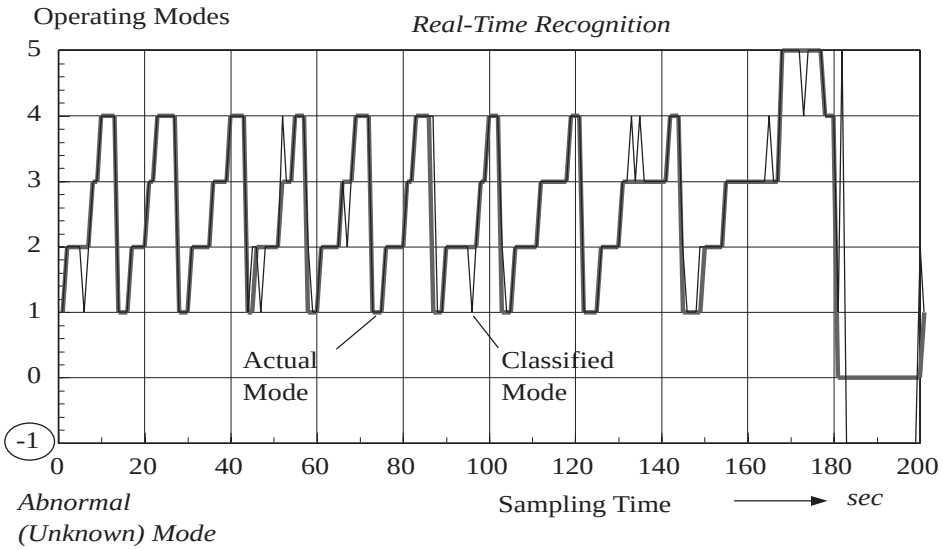

Figure 10. Results from Short-time (200 sec) Recognition of Operating.

Modes, based on 4 Parameters: $P1$, $P2$, $P5$ and $P6$.

The overall "true classification" rate of the proposed method and algorithm can be improved by several means. For example, increasing the total number of the (active) neurons in the trained SOMs leads usually (but not always) to a higher true recognition level. Some other practical considerations can also be taken into account. For example, the threshold α affects the true classification rate and needs to be selected properly, usually in an experimental way.

For comparison, the standard *One-Model* method for classification has also been applied to the same data set. When a single self-organizing map with dimensions of $LM = 11 \times 11 = 121$ neurons have been used, a true recognition level of up to *82%* has been achieved in both of the 4and the 6 parameters cases.

8. Conclusions

The above experimental results show that the proposed method and algorithm for real-time classification, based on *Separation Models* in the form of off-line trained self-organizing maps is accurate in performance

and practically applicable. The main features and advantages of the proposed method can be summarized as follows:

1. The proposed algorithm is aimed at real-time classification, based on dynamically changing real-time measured data. Therefore it uses not only the current sampling data, but also the "change-of-the data". Thus the "time tendency" which is a kind of "data trajectory" in the parameters space is used to give additional information about the characteristics of each operating mode. As a result the classification becomes more distinct and with a less degree of ambiguity.

2. In the proposed method, the notion of preliminary off-line trained *Separation Models* in the form of self-organizing maps (*SOM*) is introduced. Each *Separation Model* is trained as a separate *SOM* by using a sub-set of training data for only one operating mode. Therefore the usual labeling procedure after the end of training is not needed here.

3. The obtained *Separation Model* in this way has an advantage that it "remembers" clearly only one classification pattern and serves as a kind of "local expert" for one operating mode. The neurons of the trained *SOM* are further used as a compact set of representative points of the large number of training data for this operating mode. Such a separate training of the *SOM* has a better separation ability in the case of highly overlapped classification patterns (i.e. very close data points, but belonging to different operating modes

4. The deletion of the so called "idling neurons" at the end of the training procedure of each *SOM* is another idea implemented in the proposed method. If a certain neuron has not been classified as "winner-neuron" for any of the training data, it is considered as an "idling neuron" and is deleted from the *SOM* after the end of the training process. In this way, only really active neurons are used as representative points for the further classification.

5. The training of the *SOM* is done not at once, but as a kind of selection (optimization) procedure from several *SOM* trainings. Here the "Average Minimum Distance" $AV_{\min}$ and the "Standard Deviation" ST_{dev} of the distances between the training data points and the neurons of the *SOM* are used as parameters in the selection. As a result a *SOM* with neurons that have a better distribution among the training data points is found.

6. The proposed real-time classification algorithm has a high degree of flexibility and adaptability. This means that if a $new\,(\,Z\,+\,1\,)$ operating mode is specified by the human operator, then a collection of data for only this mode has to be obtained and used for training the *new* separation model: $SOM_{Z\,+\,1}$. After that this model is just *added* to the previously existing number of Z separation models. In such way we do not need to rebuild or re-train the entire classification model. This kind of *gradual adaptation* can continue smoothly with time when new knowledge about the operation of the machine becomes available.

References

[1] J.K. Bezdek, *Pattern Recognition with Fuzzy Membership Function Algorithms*, Plenum Pres, New York, 1981.

[2] Laurene Fausett, *Fundamentals of Neural Networks: Architecture, Algorithms and Applications*, Prentice Hall, 1993.

[3] S. Haykin, *Neural Networks – A Comprehensive Foundations*, Second Edition, Prentice Hall Inc., Upper Saddle River, New Jersey, 1999.

[4] R. Goodman, C.M. Higgins, J.W. Miller and P. Smyth, "Rule-based Neural Networks for Classification and Probability Estimation", *Neural Computing*, Vol. 14, pp. 781-804, 1992.

[5] A. Frosini, M. Gori & P. Priami, "A Neural Network-based Model for Paper Currency Recognition and Verification", *IEEE Transactions on Neural Networks*, Vol. 7, pp. 1482-1490, 1996.

[6] T. Kohonen, "Self-organized Formation of Topologically Correct Feature Maps", *Biological Cybernetics*, Vol. 43, pp.59-69, 1982.

[7] T. Kohonen, "The Self-Organizing Map", *Proc. IEEE*, Vol. 78, No. N-9, pp. 1464-1497, 1990.

[8] J.A. Kangas, T.K. Kohonen & J.T. Laaksonen, "Variants of Self-Organizing Maps", *IEEE Transactions on Neural Networks*, Vol. 1., pp. 93-99, 1990.

[9] T. Kohonen, "Self-Organizing Maps: Optimization Approaches", in *Artificial Neural Networks*, Editors T. Kohonen et al., North-Holland, Amsterdam, 1991.

[10] G. Carpenter and S. Grossberg, *Pattern Recognition by Self-Organizing Neural Networks*, Cambridge, MA: MIT Press, 1991.

[11] T. Kohonen, *Self-Organizing Maps*, 2^{nd} Edition, Springer-Verlag, 1997.

[12] E. Uchino, M. Kawamura and K. Nagata, "Dynamic Deletion of Units for Self-Organizing Map by Introducing a New Measure of Unit's Contribution to Learning", *Journal of Japan Society for Fuzzy Theory and Systems* (SOFT), Vol. 14, No. 6, pp. 157-164, Dec. 2002.

[13] Ruei-Shan Lu, Shan-Lien Lo, Diagnosing Reservoir Water Quality Using Self-Organizing Maps and Fuzzy Theory, *Water Research*, Vol. 36, pp. 2265-2274, 2002.

[14] S.-L Jamsa-Lounela, M. Vermasvuori, P. Enden and S. Haavisto, "A Process Monitoring System Based on the Kohonen Self-Organizing Maps", *Control Engineering Practice*, Vol. 11, pp. 83-92, 2003.

[15] A. Walter and K.J. Schulten, "Implementation of Self-Organizing Neural Networks for Visuo-Motor Control of an Industrial Robot", *IEEE Transactions on Neural Networks*, Vol. 4, No. 1, pp. 86-95, 1993.

[16] T.M. Martinetz, H.J. Ritter & K.J. Shulten, "Three-dimensional Neural Net for Learning Visoumotor Coordination of a Robot Arm", *IEEE Transactions on Neural Networks*, Vol. 1., pp.131-136, 1990.

[17] T.M. Martinetz and K.J. Shulten, "A "Neural-Gas" Network Learns Topologies", in Kohonen T., Makisara K., Simula O. and Kangas J., Editors, *Artificial Neural Networks*, North Holland, Amsterdam, pp. 387-402, 1991.

[18] T.M. Martinetz, S.G. Berkovich and K.J. Shulten, "Neural-Gas Network for Vector Quantization and Its Application to Time-series Prediction", IEEE Transactions on Neural Networks, Vol. 4, No. 4., pp. 558-569, 1993.

[19] T.M. Martinetz and K. J. Shulten, "Topology Representing Networks", *Neural Networks*, Vol. 7, No. 3, pp.507-522, 1994.

[20] B. Fritzke, "Growing Cell Structures – A Self-organizing Network for Unsupervised and Supervised Learning", *Neural Networks*, Vol. 7, No. 9, pp. 1441-1460, 1994.