REAL-TIME RECOGNITION OF THE OPERATING MODES OF PLANTS AND MACHINES BY USE OF SELF-ORGANIZING MAPS

Gancho Vachkov[1] and **Nikola Kasabov**[2]

[1] 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

[2] Knowledge Engineering and Discovery Research Institute, Auckland University of Technology, Private Bag 92006, Auckland1020, New Zealand;
E-mail: nkasabov@aut.ac.nz

Abstract. In this paper a new algorithm for real-time recognition and classification of different operating conditions (modes) of industrial plants and construction machines is proposed. It is supposed that the number of the operating modes is preliminary known. The algorithm utilizes the effectiveness of the Self-Organizing Maps (SOM) for creating the so called *Separation Models*, that are able to distinguish each operating condition 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 produced the least minimal distance 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.

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

1. INTRODUCTION

Many industrial plants and machines, especially the construction machines, work in different and frequently changing operating conditions (modes) rather than in a steady-state regime. Such frequent changes require a good technical condition of the machine and are more demanding from a viewpoint of reliability and maintenance conditions. In order to analyze the performance of the machine under different operating conditions and to find the beginning of a deterioration in its performance, all the operating modes should be properly distinguished and separated from each other for a further machine performance analysis.

An appropriate sensory system can be attached to the machine to measure, save and possibly transmit the readings from different parameters during the real-time operation. Then the problem of distinguishing the operating modes becomes a kind of *real-time classification problem*, which is basically more difficult than the well known off-line classification.

In this paper a new algorithm for classification and real-time recognition of different (preliminary assumed) operating modes of machines or industrial plants is proposed. This algorithm utilizes the effectiveness of the Self-Organizing Maps (SOM) for construction of the specially introduced *Separation Models*. These are models that are able to distinguish each operating mode separately, based on preliminary given experimental data.

In the proposed real-time computation procedure, at each sampling time the minimal Euclidean distances from the current measured data point to a certain node of each SOM is calculated. Then the Separation Model (represented by a respective trained SOM) that has produced the least minimal distance defines the real class of the current operating mode.

2. STATEMENT OF THE PROBLEM AND STANDARD SOLUTION OF THE CLASSIFICATION PROBLEM

In the simplest way the classification problem can be stated as follows: 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 be distributed among z classes (groups) in total: $C_1,...,C_j,...,C_z$. In the crisp classification case, each data point belongs to only one class, while in the fuzzy classification one data point may belong to different classes with different membership degree.

Then the problem is to build an appropriate model based on some logic, algorithm or heuristics that is able to assign (guess) the true class to each point. The obtained recognition model is further referred to as *classifier* of this problem.

Different learning strategies (training procedures) can be used in order to obtain the classifier. The most popular one is the "*supervised learning*" where a predefined set of $TD \leq DP$ *training* data points with a known "answer" (the actual class) is used for *off-line* training. Then at each training iteration, the classifier gradually updates its parameters.

The most often used recognition models in the standard classification problem are in the form of Self-Organizing Maps (SOM) [1,2,3] and Back-Propagation (or other) Neural Networks (BPNN) [4,5], but other classification techniques are also possible.

Figure 1. gives the idea for the standard solution of the classification problem, based on the so called *one-model* approach. Here one classifier (neural network, self-organizing map etc.) has to be trained to recognize different possible classes.

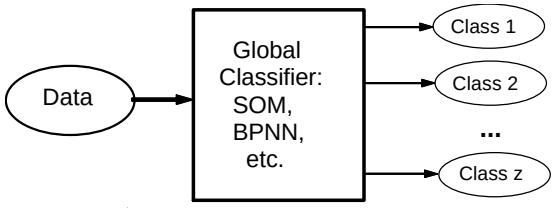


Fig. 1. The Standard *One-Model* Approach to Data Classification.

Quite often, in the real cases the available data do not carry enough information for clear distinction of the classes. It may happen that two or more data points are very close or almost coincide with each other (in the feature space) but still they belong to different classes. The following is an illustration of such difficult case for classification.

Figure 2. shows experimental data, taken by sampling of two parameters (*P1* and *P2*) of a hydraulic excavator. The purpose is to use these data for distinction (classification) of the following *4* operating modes of the excavator:

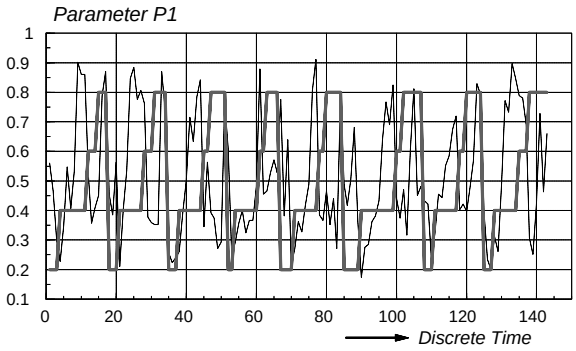
Mode 1. Loading the bucket with the raw material (sand, stones etc);

Mode 2. Moving the load to the truck;

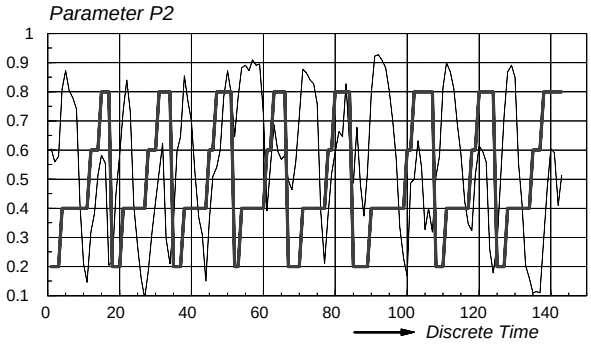
Mode 3. Unloading the bucket material into the truck;

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

The operating modes are shown in Fig. 2. as bold lines with *4* different steps, which follow periodically 8 times in the sequence: *1,2,3,4*. These data is supposed to be further used for a supervised learning of the recognition model. However they do not provide good information for clear distinction of all four modes. The problem is displayed graphically in Fig. 2a) and 2b), where an overlapping in large area of the parameters space is noticed. Therefore it is expected to be quite difficult and not reliable for one model only (i.e. one classifier) to distinguish all operating modes clearly.

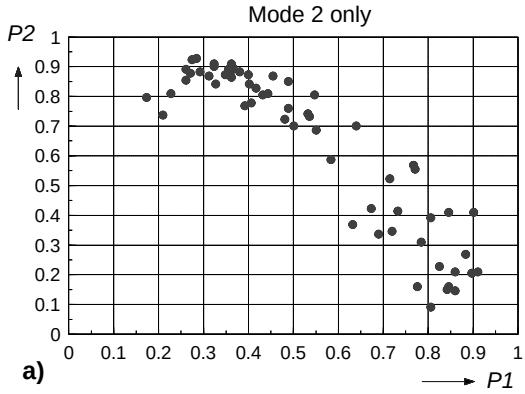


a)



b)

Fig. 2. Measured Data from 2 Parameters for Classification of 4 Operating Modes



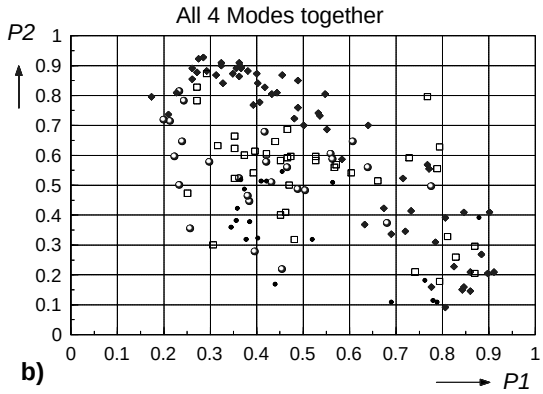


Fig. 3. *a)* Plot of the Data for one Mode;
b) Overlapping of the Data for All *4* Modes

The self-organizing maps (SOM) are further chosen in this paper as an appropriate structure for creating the so called *separation models* in the new proposed classification algorithm. Therefore the SOM structure as well as the training algorithm for the self-organizing maps is explained in the next section.

3. STRUCTURE AND COMPUTATIONAL PROCEDURE FOR TRAINING OF THE SELF-ORGANIZING MAP

The self-organizing feature maps, first proposed by Kohonen [1,2,3] belong to the group of network models that use unsupervised learning, i.e. they are able to organize a set of input pattern into classes independently. Nevertheless they can be also successfully used for supervised learning as in [7,8], if preliminary information exists about the classes. Then after the training, the self-organizing map is used as a classifier of the new obtained data. It actually assigns each input data point to the respective (most plausible) class.

There are some variations [2,3,6] among the general structure and the computational algorithms for training of the self-organizing maps. Further on we keep the general scheme with some modification in the training algorithm as in [6].

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 these 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. 4. Thus the total number of all neurons is $LM = L \times M$ and normally $TD >> LM$.

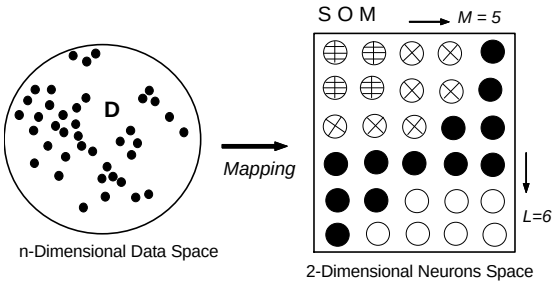


Fig. 4. Example of Mapping the *n*-Dimensional *Data Space* onto 2-Dimensional *Neuron Sp*ace with *30* neurons and *4* Classes.

Each neuron is referred to in the two-dimensional space by its *spatial coordinates* $N(i, j),\ i = 1, 2, ..., L;\ j = 1, 2, ..., M$. Then the *Euclidean* distance *d(i,j)* between a pair of neurons *N(i,j)* and *N(p.q)* is calculated as in [6]:

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

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:

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

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)*:

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

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 preliminary determined and the initial weights (3) for all *LM* neurons should be 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 the improper (too low or too big) number of *T* may cause premature stop or exceeded training that leads to a improper classification performance.

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

$$w_{t+1}^r(i, j) = w_t^r(i, j) + \Delta w_t^r(i, j)\ ,\ r = 1, 2, ..., n. \tag{5}$$

where:

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

Here the *learning rate*:

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

is chosen as a monotonically decreasing function (linear in this case), as in [3,6], in order to guarantee the final convergence of the learning process.

The N*eighborhood Area Function* $R_t(i, j)$ in (6) is used to control the *spatial property* of the self-organized network. It means that neurons, which are closer to each other, receive more learning increment than the far located neurons.

Among the big variety for neighborhood area functions, further on we use the following formula, proposed in [6]:

$$R_t(i,j) = exp[-b_t d^2(i,j)], \quad (8)$$

$$b_t = b_o (t/T)^s \quad (9)$$

Usually *s = 2,3* or *4*. Here $\delta(i,j)$ is the 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 closest located neuron to the current data point *d*(*h*) in the *n*-dimensional space. The minimum distance:

$$j_h(p.q) = \min_{\substack{1 \le i \ge L \\ 1 \le j \ge M}} \{ j_h(i,j) \} \quad (10)$$

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 they such "neighbors" gradually gather closer to the "winner", thus forming a kind of cluster. In addition, the size of the "neighborhood" is gradually decreasing (linearly in this case) in order to ensure the convergence and the generalization ability of the network.

A graphical illustration in Fig. 5. shows how the different tuning parameters influence the learning rate during the training of the SOM.

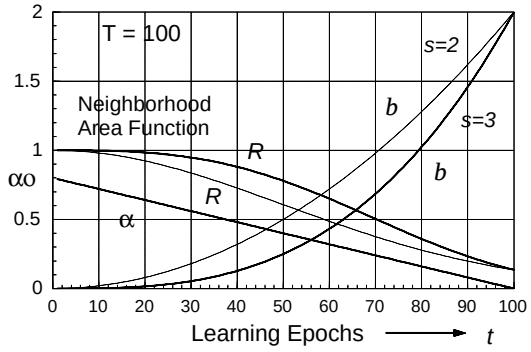


Fig. 5. Influence of the Different Parameters to the Learning Rate of the Self-Organizing Map

One important goal of the training process is to obtain a "good distribution" of the neurons among the "cloud" of all training data points in the high-dimensional data space. As a measure of the distribution level, we propose here the *Average Minimum Distance* $AV_{\min}$ by taking the average from all *Minimum Distances MD(i), i = 1,2,…,TD*:

$$AV_{\min} = \frac{1}{TD} \sum_{i=1}^{TD} MD(i) \quad (11)$$

Here the *Minimum Distance MD(i)* for the *i*-th data point *d(i), i = 1,2,…,TD* is the Euclideam distance between this point and its respective "winner-neuron" *n(j)* as explained above by equation (10).

In our off-line training procedure, we assume that "*best trained*" Self-Organizing Map is that one with the least *AVmin* obtained through a kind of *Multi-training Optimization* procedure.

In addition, another idea is implemented in our off-line training procedure, namely: the neurons, which haven't contributed to the process of finding the minimal distance are labeled as "*idling neurons*" and are deleted from the SOM model after the training.

By our definition, a neuron is categorized as "*idling neuron*", if it has not been a "winning-neuron" for any of the all *TD* data points.

The following Fig. 6. illustrates graphically the minimum distances *MD* between *10* data points and *7* neurons in a SOM. It also gives an llustration of two "idling neurons: *n(3)* and *n(7*) is also depicted.

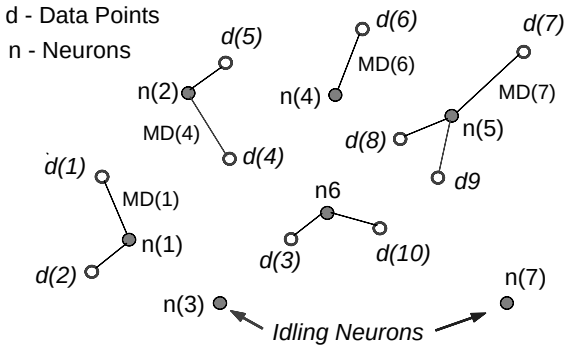


Fig. 6. Representation of the Minimum Distances *MD(i)* between *10* Data Points and *7* Neurons on the Self-Organizing Map (SOM).
Two Neurons: *n(3)* and *n(7*) have been defined as "*Idling Neurons*".

After the training, the SOM consists of neurons that are located most close (in average) to the data points, thus minimizing the average distance to them. In addition, these neurons are more concentrated to the areas with bigger concentration of 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.

4. THE PROPOSED REAL-TIME CLASSIFICATION ALGORITHM

The new proposed algorithm for classification and real-time recognition of different (preliminary assumed) operating modes utilizes the effectiveness of the Self-Organizing Maps (SOM) for creating the specially introduced *Separation Models*.

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 clearly represent this operation mode only. In such way, the separation Model SOM_i serves as a "*Local well trained Expert*" which is able to clearly distinguish the *i*-th operation mode only. 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 newly proposed algorithm for recognition of the operating modes is depicted in the general block-diagram shown in Fig.7.

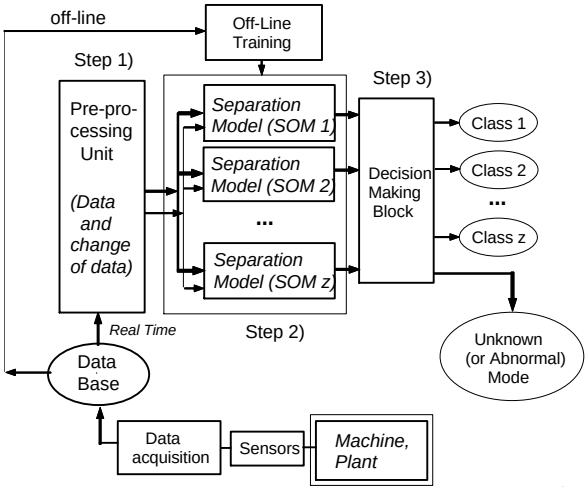


Fig. 7. General Flowchart of the Proposed Algorithm for Real-Time Recognition of Machine *Operating Modes*

As seen, there are two main Computation Steps in the proposed idea, as follows: *Off-line Step* and *Real-time Step* that use the *off-line data stream* and the *real-time data stream* respectively.

A) *Off-line* Computation step. This is a preliminary computation at which *off-line* training of the Separation Models: SOM_1, SOM_2, …, SOM_z is performed. For this purpose a large and reliable experimental data for each Operation Mode have to be obtained beforehand. These data are stored in a Data Base and then used for off-line Training of each Self-Organizing Map: SOM_1, SOM_2, …, SOM_z. The training procedure for the separation models, explained in Section 3. is usually fast one, since less training data (for one mode only) are used: TD_i, $i = 1,2,\ldots,z$, instead of all *TD* data. Therefore generally, less number of neurons LM_i, $i = 1,2,\ldots,z$ will be also 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.

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

As shown in the above Fig. 7., the real-time classification procedure consists of the following three calculation sub-steps:

Step B.1) *Preprocessing Step*. Here after the dynamical 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 for the previous sampling *(k-1)*. In such way, the input data pattern is created for this sampling time.

The motivation for such additional calculation procedure, that defines the differences *Dd(k)* is the fact that we are dealing with dynamical data. Therefore only the current measuring does not give sufficient representative information about the dynamics of the whole process. By taking into account the differences, we can "catch" the *tendency* in data trajectories, which could be unique for each operation mode at one sampling time. Obviously such additional data parameter *Dd(k)* in the calculation step means that we have to create also a SOM with double size: *d(k)* and *Dd(k)*, which will require more training time. However it is done once in off-line mode and do not possess any practical problems. The gain which we obtain is: increasing of the "true" classification cases during the real-time recognition.

Step B. 2) "*Similarity Degree*" *SD(k)* calculation between the current input pattern: *{d(k);Dd(k)}*, and each of the Separation Models (the SOMs). This is calculated as the minimum distance between the current observed pattern {*d(k);Dd(k)}* and a node *n(j)* of within the same Separation Model.

Among the different possible ways for calculation of such similarity degree, here again we use the Euclidean distances [2,3,6] to define the closest neuron to the observed input pattern {*d(k);Dd(k)*}. Please note that the notion of Similarity Degree *SD(k)* is similar to the notion of Minimum Distance *MD(i)* in (11) from a calculation viewpoint. The difference is that *SD(k)* is applied to a current measured (not seen before) data point and all possible separation Models, while the Minimum Distance *MD(i)* is applied to a training point *d(k)* and all neurons in one SOM.

From a physical point of view, the least value of *SD(k)* represents the highest similarity degree for the current data point *d(k)*.

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);Dd(k)}*. This step can be performed in different ways, which directly influence the result of classification. One, probably the simplest implementation of this step is as follows:

The *r*-th Separation model SOM_r that shows the highest similarity degree, namely:
$SD(r) = min\{SD(i)\}$, $i=1,2,..,z$ defines the type of the most plausible operating mode for the current input

data pattern.

5. EXPERIMENTAL RESULTS AND CONCLUSIONS

The proposed idea and algorithm for real-time classification of the operating modes, based on the separation models is investigated on many test examples as well as on real experimental data from a hydraulic excavator for the same *4* operating modes, mentioned in Section 2. Firstly, the recognition procedure has been computed by using dynamical data from *2* parameters only (Engine Speed and Boost Pressure) that have been shown in Fig. 2. The results of recognition are displayed in Fig. 8. The bold broken line corresponds to the actual (true) operating modes, while the thin line represents the recognized modes. The total result is: *89.5%* true classification.

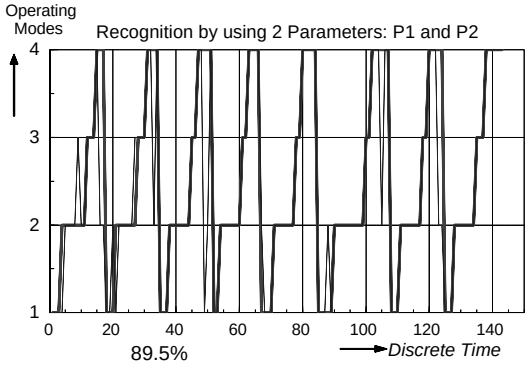


Fig. 8. Classification of the Operating Modes by Use of Two Parameters.

In Fig. 9. even better classification level of *97.2%* is achieved, with all *6* measured parameters being included in the classification procedure. The additional parameters: *P3*,*P4*,*P5* and *P6* supply more detailed information about the operating condition of the excavator, namely the hydraulic pumps pressure, engine oil pressure and fuel consumption. Therefore the improved classification results are quite reasonable.

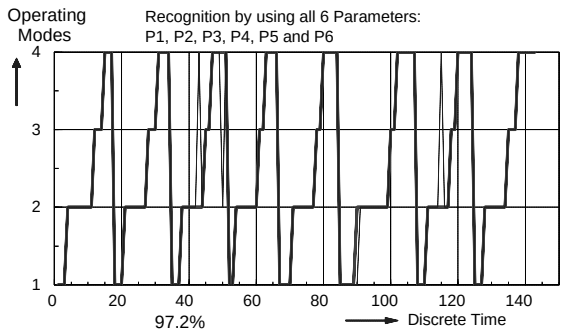


Fig. 9. *Classification of the Operating Modes for the all 6 Parameters*

Finally, the performance of the propose classification algorithm has been compared to the results from the standard: *One-SOM-model* approach. When a single Self-Organizing Map with big dimension of *LM = 11*x*11 = 121* neurons have been used, a true recognition level of less than *80%* has been achieved in both: *2* and *6* parameters case.

These experimental results show clearly that the proposed algorithm for classification, based on the idea of creating Separation Models in the form of self-organizing maps, is flexible and accurate in performance, with ability for practical and real-time application.

REFERENCES

1. T. Kohonen, Automatic Formation of Topological Maps of Patterns in a Self-Organizing System, *Proceedings of the 2nd SCIA, Scand. Conference on Image Analysis*, Helsinki, Finland, 214-220, 1981.

2. T. Kohonen, The Self-Organizing Map, Proc. IEEE, vol. 78, No. N-9, 1464-1497, 1990.

3. T. Kohonen, Self-Organizing Maps, 2nd Edition, Springer-Verlag, 1997.

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

5. R. Goodman, C.M. Higgins, J.W.Miller and P. Smyth, Rule-based Neural Networks for Classification and Probability Estimation, *Neural Computing*, vol. 14, 781-804, 1992.

6. 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, 157-164, Dec. 2002.

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

8. 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, 83-92, 2003.