A Self-Organizing Neurocontroller for Vibration Suppression

Dimitrios Moshou[†], Jan Anthonis, Pál Jancsók, Herman Ramon
Department of Agro-Engineering and Economics, Laboratory for Agro-Machinery
and Processing, K.U. Leuven, Kardinaal Mercierlaan 92, Blok E,
3001 Heverlee, Belgium

Abstract

The Self-Organizing Map Neural Network is used in a supervised way to represent a sensor-actuator mapping. The learning of the controller assumes no prior information, but only reward/failure signals that are produced by an evaluation criterion. The evaluation criterion used is based on the low-pass filtering of the gradient of a reward function and the local storing of the filtered gradient value. The control method is tested in vibration isolation of a flexible spray boom used in agriculture for pesticide application. The Neural Network learns to stabilise the boom on-line without any prior information and with a very high performance.

1 Introduction

Many mechanical structures are subjected to vibrations that can lead to damage or to fatigue and thus shorten the operational lifetime of the structure. Passive vibration isolation gives poor results because of low selectivity. Active vibration isolation is much more performant. However, model-based techniques require persistent excitation signals. In practice persistent excitation is rarely available during the operating condition of a system. An alternative is to develop an algorithm that can discover the control actions by itself. The only source of information in this case is a “reward function” which specifies at a given moment how well the controller has performed. For this algorithm to be executed, the system must now create at each learning step the control action, as shown in (Ritter *et al.*, 1992). In the absence of any further information a stochastic search can be performed in the space of the available control values with the aim to maximize the reward received at each step. Performance based partitioning of the state-space is achieved. Current sensor, actuator and target sensor values in a vectored form become associated with next step control actions derived from the maximisation of a certain reward function.
For continuous state-spaces, the state-action look-up table refers to a quantization of the states of the system through the use of an adaptive algorithm. Basically two types of learning are present here:

i) the adaptation of the partitioner, and
ii) the reinforcement learning of the controller, as shown in (Hermann and Der, 1995).

The determination of quantized states, which are internal states in the full control problem, represents an instance of the hidden state problem, as shown in (Das and Mozer, 1994). For discrete actions, an ideal

---

[†] Email: *dimitrios.moshou@agr.kuleuven.ac.be*

partition of the continuous state space consists of domains with each having a unique optimal action for all states belonging to that domain. In this way, an optimal partition is defined by the use of a policy function that assigns states to actions. In the current paper, the partitioning is performed by using a learning vector quantizer, as shown in (Ritter *et al.*, 1992). In such a case, the cells are defined by reference vectors in the input space. The distribution of the reference vectors is usually determined by statistical properties of the vector quantizer's inputs, as shown in (Ritter and Schulten, 1986). Thus, around the stable states of the controlled system, fine-grained partitions are formed.

In the present paper, the learning rule for the vector quantizer is based on Kohonen's Self-Organizing Feature Map, as shown in (Kohonen, 1995), which possesses interesting noise filtering properties. The Self-Organizing Map commonly referred to as SOM, first presented in (Kohonen, 1982), is a neural network (NN) that converts complex, nonlinear statistical relationships between high-dimensional data into simple geometric relationships. The determination of quantized state cannot by itself represent input-output relationships. By extending the SOM with output weights that store the output part of a mapping can provide the original algorithm with the ability to approximate continuous relationships. Such a network has been introduced earlier, in (Ritter *et al.*, 1992). In the current paper, for the first time, a partitioner based on the SOM is learned simultaneously with a reinforcement signal based learning controller. A novel training algorithm is presented for updating the parameters of this network. Then, this training algorithm is successfully applied in the on-line stabilisation of a flexible agricultural spray boom.

2 Controller and Partitioner Learning

The Self-Organizing Map is a neural network (NN) that maps signals (**x**) from a high-dimensional input space (V) to a one- or two-dimensional discrete lattice of neuron units (A). Each neuron stores a weight ($\mathbf{w}_s$). The map preserves topological relationships between inputs in a way that neighbouring inputs in the input space (V) are mapped to neighbouring neurons in the map space (A). When extended with output weights ($\mathbf{y}_s$) it can learn in a supervised way the input-output (I/O) mapping $\mathbf{y} = \mathbf{f}(\mathbf{x})$, where y belongs to the output space (U). This association is shown in Figure 1.

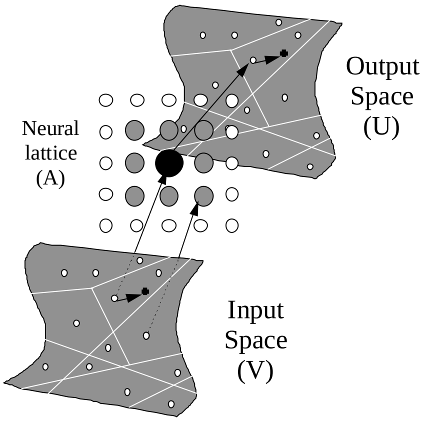

Figure 1. Association of input-output values by using a Self-Organizing Map

The association of situation-action pairs is based on a reward/punishment scheme where a reinforcement signal is produced. A two-state reinforcement signal (+1 or 0) is produced after an evaluation of the result that a certain action has brought. In a control situation when a system is driven by a certain control sequence:

$$\mathbf{u} = (u(k-1),u(k-2),...,u(k-n))^T \tag{1}$$

And the response of the system is measured:

$$\mathbf{y} = (y(k),y(k-1),...,y(k-n))^T \tag{2}$$

where the desired next step output is denoted as $\mathbf{y}_d$ (generally a vector, but in the example with the boom it is assumed to be a scalar), the state vector that is used as input to the state quantizer (SOM) is constructed as follows:

$$\mathbf{x} = (\mathbf{y}_d^T, \mathbf{y}^T, \mathbf{u}^T)^T \tag{3}$$

Subsequently, these state vectors are clustered by the SOM. The control values $\mathbf{u}(k)$ are stored as an output weight through the training procedure of Kohonen's algorithm, as shown in (Kohonen, 1995). In the case of MIMO systems, the data can be concatenated in the same vector. The use of SOMs to cluster concatenated sequential data has already been attempted by Kangas (1990). A scalar reward function that determines the association of states to actions can be defined as

$$R(k) = -(\mathbf{y}(k)-\mathbf{y}_d)^T Q\,(\mathbf{y}(k)-\mathbf{y}_d) \tag{4}$$

Where, with $\mathbf{y}(k)$ the vector of current (at t=k) output measurements is defined, $\mathbf{y}_d$ is the vector of current target output values, and Q is a positive definite matrix (in the flexible boom application Q is a unity matrix). A policy can be based on this reward function by calculating the difference between two consecutive values of the reward function, thus approximating the first derivative of the reward function with respect to time (T_s is the sampling period):

$$\frac{\Delta R}{T_s} = \frac{R(k) - R(k-1)}{T_s} \cong \dot{R} \tag{5}$$

In Figure 2 the evaluation is performed by the look-up table block and produces a firmness signal that modifies the state quantizer. It is clear that every increase of ΔR is desirable, since the maximum target value of the reward function (R) is zero. However, a maximisation of R over a number of steps is better because temporary variations of the reward function can be due to disturbances and not caused by the control sequence. For this reason every neuron stores a moving average of the increase ΔR of the reward function R. This moving average is denoted as $\langle \Delta R \rangle$.

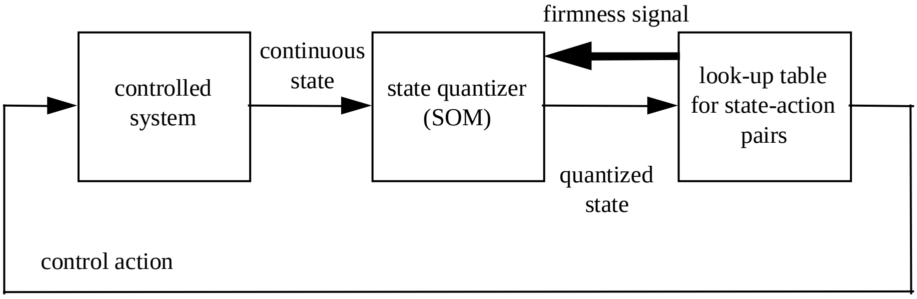

Figure 2. Scheme of learning problem of combined controller and partitioner learning

In the presented method when a neuron is selected and a control action produces an increase over the stored moving average for this specific neuron, a positive reinforcement signal is produced indicating that this neuron and its immediate neighbours are allowed to learn the state-action pair that has led to the positive reinforcement signal. But since only increases of the reward function that are greater than the stored moving average for each neuron lead to learning, a continuous improvement of the partitioner and the look-up table of control-action pairs is achieved. The moving average of the increases of the reward function can be obtained for time-step t=k:

$$\langle \Delta R \rangle_k = \langle \Delta R \rangle_{k-1} + \gamma(\Delta R_k - \langle \Delta R \rangle_{k-1}) \tag{6}$$

where γ is a small positive constant. Note that the momentary value of the increase is denoted without brackets. After the update, the new moving average is stored in the activated neuron. The whole concept has to do with supplying to each neuron a bias term to avoid overtraining. The learning algorithm for the input and output weights is derived from the original Kohonen algorithm, as shown in (Kohonen, 1982):

$$\Delta \mathbf{w}_s^{(in)} = \varepsilon h(\mathbf{x}-\mathbf{w}_s^{(in)}) \tag{7}$$

$$\Delta \mathbf{w}_s^{(out)} = \varepsilon' h'(\mathbf{u}-\mathbf{w}_s^{(out)}) \tag{8}$$

Where ε, ε' and h, h′ are the learning rates and the neighborhood kernels respectively. With $\mathbf{w}_s^{(out)}$ the output weight $\mathbf{y}_s$ is denoted. It must be noted that in the updating equations the winning neuron is denoted with **s**, i.e. the one that has the smallest distance from the input **x**. However the updating equations apply to the lattice neighbours of the winning neuron at every updating step. The neighbourhood kernels used have the form of a Gaussian distribution like:

$$h = \exp(-\|\mathbf{x}-\mathbf{w}_s\|^2/\sigma^2) \tag{9}$$

Where $\|.\|$ denotes the Euclidean norm and σ denotes the variance of the Gaussian distribution. In either case, the applied control action that is applied, is constructed by two components:

$$\mathbf{u}(k) = \mathbf{w}_s^{(out)} - a_s(\mathbf{y}(k)-\mathbf{y}_d) \tag{10}$$

Where a_s is a small positive value (reduced slowly to a final much smaller value). This allows rapid improvement initially and allows a small margin for adaptation after the partitioner and look-up table have been learned. The value of control action u(k) from equation (10) is used for updating the output weight of updating equation (8) only in case the reinforcement signal is positive. Such updating of the output weights ensures that the controller improves continuously. The updating equation for the gain factor a_s is:

$$\Delta a_s = \varepsilon'' h''(a - a_s) \qquad (11)$$

In equation (11), ε'' and h'' are the learning rate and the neighborhood kernel respectively. By setting ε'' very small (in the example of the boom it is set equal to 0.005) the "exploration step" will converge to the final value denoted as (a) slowly enough to allow for satisfactory learning of control actions. If the final value (a) is set different than zero some residual plasticity is allowed.

Alternative ways of producing the control action of equation (10) include the utilisation of Local Linear Mappings as shown in (Moshou *et al.,* 1997), or "fuzzy" interpolation of outputs of neighbouring neurons.

3 Flexible Boom Stabilisation

Flexible spray booms are used in the agricultural domain for pesticide application. They usually consist of lightweight beams on which spraying nozzles are mounted. When driving a tractor over a field, the unevenness of the soil causes the flexible boom to vibrate, leading to under- and over-application of pesticides, thus resulting in environmental pollution. Stabilisation of flexible spray booms is needed in order to achieve a uniform spraying liquid distribution and avoid environmental damage. The learning algorithm of section 2 is applied in the on-line vibration isolation of a 12th order linearised model of a flexible spray boom (total length of 12m tip to tip). The test set-up from which the model has been obtained is shown in Figure 3. The linearised model is of the form:

$$\begin{aligned} \dot{x} &= Ax + Bu + Ew \\ y &= Cx + Du + Fv \end{aligned} \qquad (12)$$

The E matrix is the disturbance input (w) distribution matrix and v represents sensor noise. The direct feedback (D) matrix appears because of the collocation of the sensor and the actuator (y, u are both accelerations). For the simulations that are presented, only translational motion in the horizontal plane of the flexible boom is considered (SISO system). In Figure 3 is shown that the horizontal acceleration caused by translational motion is measured by the accelerometer that is attached on the boom.

The disturbances used are the accelerations resulting from a standardised field track as shown in (Norme Internationale, 1979), fed through a model of the tractor wheels and a model of a tractor on which the spray boom has been attached. The excitation signal runs for 23 sec and is shown in Figure 4. The tractor was supposed in the model to run with a constant speed of 5 km/h. The excitation signal of Figure 4 is used as a disturbance input (w) to the system. The system is discretised (Tustin transform) and integrated with Runge-Kutta of fifth order. This method of simulation is preferred because different sampling rates are used, and so effects of latency can be assessed realistically. The uncontrolled response of the system is shown in Figure 5. The sensor (accelerometer) and electro-hydraulic actuator are supposed to be collocated at 0.25m from the connection joint of boom. The input of the network consists of vectors of delayed input and output values of the system (accelerations) of a certain length (in this case two, and y_d=0):

$$\mathbf{x} = (y_d, y(k), ..., y(k-n), u(k-1), ..., u(k-m))^T \qquad (13)$$

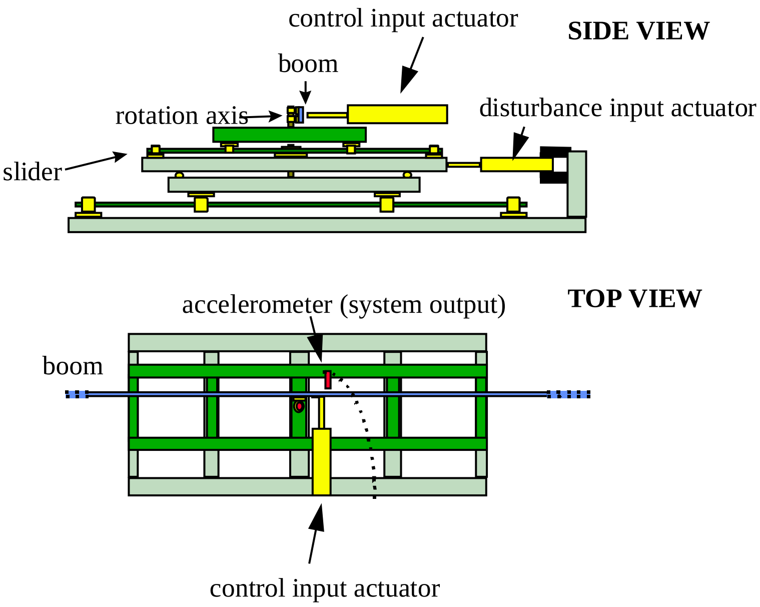

Figure 3. Set-up for spray-boom measurements (sensor and actuator collocated)

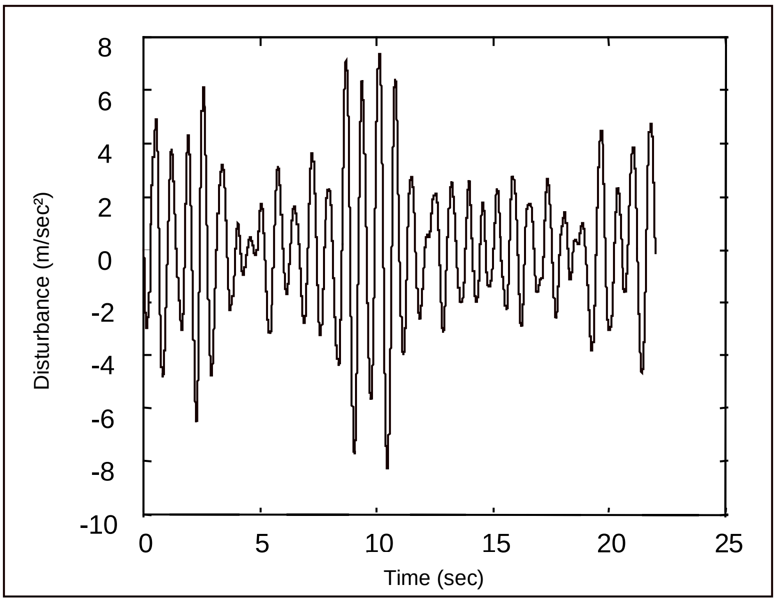

Figure 4. The acceleration profile of the standardised track used as input to the flexible boom

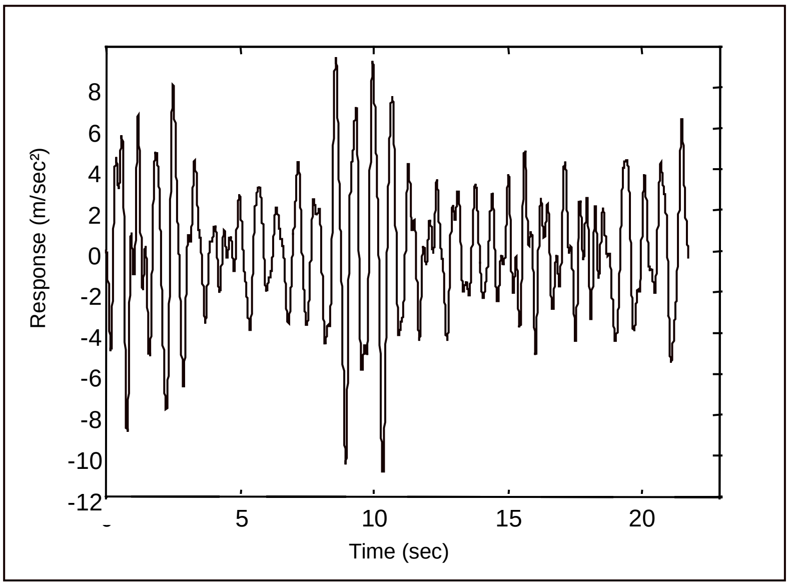

Figure 5. The response of the system when excited by a standardised track

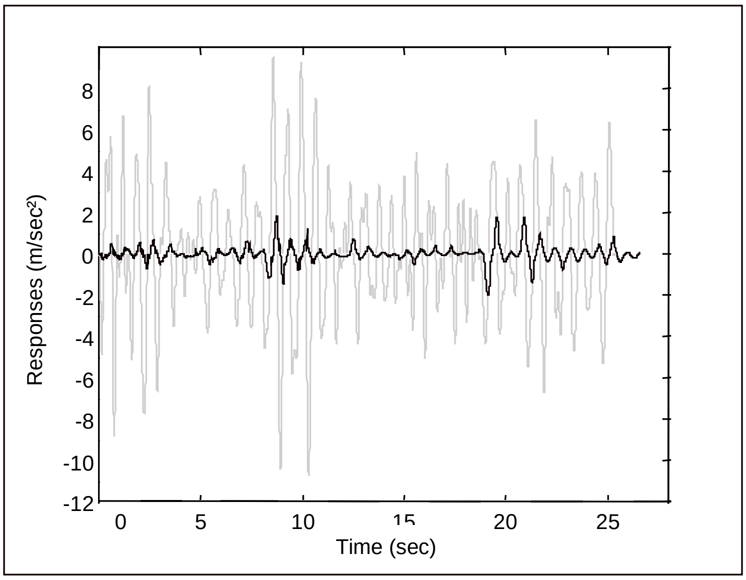

Figure 6. Controlled against uncontrolled system response (thick line: controlled)

The entire systematic procedure of section 2 has been followed. The values of the parameters (ε, σ) in the updating equations (7), (8) and (11) were chosen to decrease exponentially with time from a large initial to a small final value. The result of following the above on-line learning of control actions is shown in Figure 6. From Figure 7 it is evident that the peaks have been reduced by 20 dB. The evolution of the reinforcement signal through the first sampling steps (at a sampling rate of 1 kHz) is shown in Figure 8. Most of the learning occurs during the first 0.2 sec, thus resulting in a very small acceleration from the very beginning of the training session.

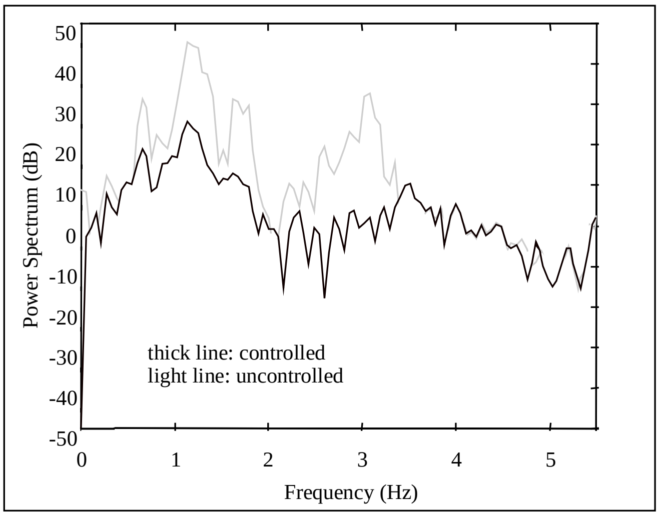

Figure 7. Controlled against uncontrolled frequency response of the system

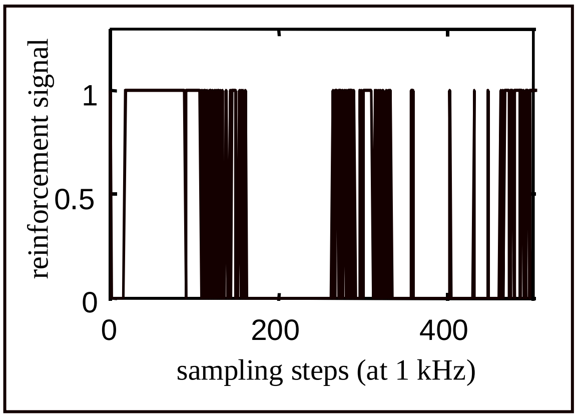

Figure 8. The reinforcement signal during the initial 0.5 sec of the training session

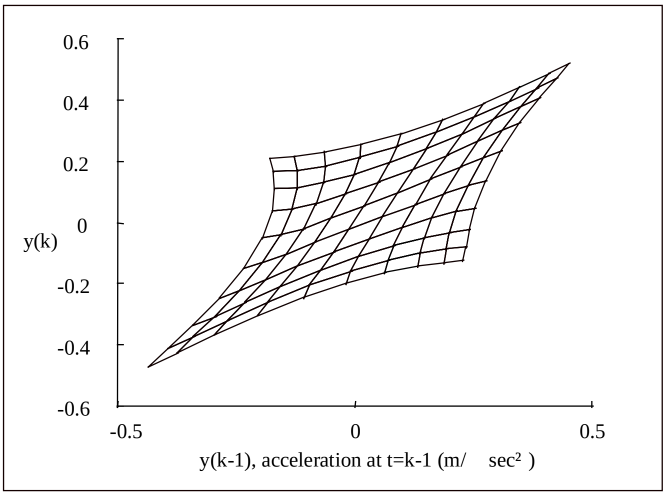

Figure 9. The SOM grid at the end of the learning period

A way of visualising the spatial structure of the representative vectors that the Self-Organizing Map has stored is by plotting these vectors in the case that they are also two-dimensional like the SOM itself. In the case of a higher dimension of the input data the geometrical relations of the representative vectors are difficult to visualise. As is evident from Figure 9, in which the first two weights of the map are plotted, there is a very clear ordering at the end of learning. These two weights are representative values of two consecutive controlled acceleration values that have emerged through the learning process of the SOM. It has to be mentioned that during learning the SOM tends to represent the states that occur more frequently. However because of the bias that is introduced through the moving average of the reward function increase in the updating policy the states that are visited tend to be equiprobable. An important aspect of the final stage of the SOM is that the states around the diagonal cover a wider range. This follows from the cooling schedule of the updating equations (7), (8); i.e. the learning rate assumes a very small final value.

4 Conclusions

A new neural network method for disturbance suppression of dynamical systems has been presented. The main advantage of this method is the local representation of the controller and state partitioner which are learned simultaneously by reward and failure signals. The whole learning scheme doesn’t need any prior information but only output measurements of the controlled system. Local updating algorithms assure much faster convergence than global updating algorithms. The method is generally applicable from the point of view that it is not based on a model of the system under control. It only relies on a reward function and the moving average of locally stored rewards over time. It can be used equally well for on-line control of linear and nonlinear systems or systems with changing parameters. The new method presented can be applied in the automotive (vehicle suspensions) and the aerospace domain (flexible space structures), in systems with uncertain or complex dynamic behaviour.

References

Das, S. and M.C. Mozer (1994). "A unified gradient-descent/clustering architecture for finite state machine induction," in *Advances in NIPS 6* (J.D. Cowan, G. Tesauro, J. Alspector, Eds.), M.I.T. Press, pp. 19-26.

Hermann, M. and R. Der (1995). "Efficient Q-Learning by Division of Labour," in *Proceedings ICANN ‘95* (EC2 & Cie ed.), Paris, **2**, pp. 129-134.

Kangas, J. (1990). "Time-Delayed Self-Organizing Maps," in *Proceedings of the IEEE IJCNN-90 Conference,* S. Diego, CA, **2**, pp. 331-336.

Kohonen, T. (1982). "Self-Organized Formation of Topologically Correct Feature Maps," *Biological Cybernetics,* **43**, pp. 59-69.

Kohonen, T. (1995). *Self-Organizing Maps,* Springer Series in Information Sciences, Springer- Verlag, Berlin.

Norme Internationale, ISO 5008 - (1979) (F). *Tracteurs et Matériels Agricoles à Roues- Mesurage des Vibrations Transmisses,Globale-ment au Conducteur.* Organisation Internationale de Normalisation.

Ritter, H., T. Martinetz, and K. Schulten (1992). *Neural Computation and Self-Organizing Maps: An Introduction,* Addison-Wesley, New York.

Ritter, H. and K. Schulten (1986). "On the stationary state of Kohonen’s self-organizing sensory mapping," *Biol. Cybernetics,* **54**, pp. 99-106.

Moshou, D., J. Anthonis, and H. Ramon (1997). "Extended Self-Organizing Maps for Function Approximation and System Identification," in *Proceedings WSOM'97, International Workshop on Self-Organizing Maps*, Helsinki, Finland, pp. 181-186.