Engineering Applications of the Self-Organizing Map

TEUVO KOHONEN, FELLOW, IEEE, ERKKI OJA, SENIOR MEMBER, IEEE,
OLLI SIMULA, SENIOR MEMBER, IEEE, ARI VISA, SENIOR MEMBER, IEEE, AND JARI KANGAS

Invited Paper

The self-organizing map (SOM) method is a new, powerful software tool for the visualization of high-dimensional data. It converts complex, nonlinear statistical relationships between high-dimensional data into simple geometric relationships on a low-dimensional display. As it thereby compresses information while preserving the most important topological and metric relationships of the primary data elements on the display, it may also be thought to produce some kind of abstractions. These two aspects, visualization and abstraction, occur in a number of complex engineering tasks such as process analysis, machine perception, control, and communication.

The term self-organizing map signifies a class of mappings defined by error-theoretic considerations. In practice they result in certain unsupervised, competitive learning processes, computed by simple-looking SOM algorithms.

The first SOM algorithms were conceived around 1981–1982, and the popularity of the more advanced SOM methods is growing at a steady pace. Many industries have found the SOM-based software tools useful. The most important property of the SOM, orderliness of the input-output mapping, can be utilized for many tasks: reduction of the amount of training data, speeding up learning, nonlinear interpolation and extrapolation, generalization, and effective compression of information for its transmission.

I. INTRODUCTION

A. Objectives

The self-organizing map (SOM) is a neural-network model and algorithm that implements a characteristic nonlinear projection from the high-dimensional space of sensory or other input signals onto a low-dimensional array of neurons. The SOM is able to map a structured, high-dimensional signal manifold onto a much lower-dimensional network in an *orderly* fashion. The mapping tends to preserve the *topological relationships* of the signal domains. Due to this order, the image of the signal space tends to manifest *clusters* of input information and their relationships on the map.

Accordingly, the most important applications of the SOM are in the *visualization* of high-dimensional systems and processes and discovery of *categories* and *abstractions* from raw data. The latter operation is called the *exploratory data analysis* or "data mining."

In engineering, the most straightforward applications of the SOM are in the identification and monitoring of complex machine and process states, otherwise very difficult to perceive and interpret. The SOM can also be utilized for the development of new pattern classification and target recognition systems, whereby categorization of the input signal states is performed by it. In combination with reinforcement learning, the SOM can be effectively used to implement various control functions. Moreover, the ordered signal mapping has been found very useful for certain tasks in telecommunications, such as increasing error tolerance in signal detection and image transmission.

B. The SOM Algorithm

The SOM may be described as a nonlinear, ordered, and smooth mapping of high-dimensional input data domains onto the elements of a regular, low-dimensional array. A physical analogy of the SOM is an *ordered decoder array* in which one and only one of the decoders responds to a particular domain of input signals. It is the location of the decoder and not the exact magnitude of its response that characterizes input information. Unlike in digital-computer technology, however, the input signal sets can usually be regarded as real vectors in a metric space, typically a high-dimensional Euclidean space.

This mapping is implemented by the SOM algorithm in the following way. Assume that the set of input variables $\{\xi_j\}$ is definable as a real vector $\mathbf{x} = [\xi_1, \xi_2, \cdots, \xi_n]^T \in \Re^n$. With each element in the SOM array we associate a parametric real vector $\mathbf{m}_i = [\mu_{i1}, \mu_{i2}, \cdots, \mu_{in}]^T \in \Re^n$. The "image" of an input vector $\mathbf{x}$ on the SOM array

Manuscript received October 1, 1995; revised March 14, 1996.
The authors are with Helsinki University of Technology, Neural Networks Research Centre, FIN-02150 Espoo, Finland.
Publisher Item Identifier S 0018-9219(96)07176-9.

is defined by the decoder function: assuming a general distance measure between $\mathbf{x}$ and $\mathbf{m}_i$ denoted $d(\mathbf{x}, \mathbf{m}_i)$, the image is supposed to have the array index c defined as

$$c = \arg\min_i \{d(\mathbf{x}, \mathbf{m}_i)\}. \quad (1)$$

Our task is to define the $\mathbf{m}_i$ in such a way that the mapping is ordered and descriptive of the distribution of $\mathbf{x}$.

One may try to determine the $\mathbf{m}_i$ by an optimization process, following the idea of the classical vector quantization (VQ) [49], [53], [114]. In the latter, a finite set of *codebook vectors* $\{\mathbf{m}_i\}$ is placed into the space of the $\mathbf{x}$ signals to approximate them. Let $p(\mathbf{x})$ be the probability density function of $\mathbf{x}$, and let $\mathbf{m}_c$ be the codebook vector that is closest to $\mathbf{x}$ in the signal space, i.e., $d(\mathbf{x}, \mathbf{m}_c)$ is smallest. The VQ shall then minimize the *average expected quantization error function*

$$E = \int f[d(\mathbf{x}, \mathbf{m}_c)]p(\mathbf{x})d\mathbf{x} \quad (2)$$

where f is some monotonically increasing function of the distance d.

Notice that the index c is also a function of $\mathbf{x}$ and all the $\mathbf{m}_i$, whereby the integrand of E is not continuously differentiable: c changes abruptly when crossing a border in the signal space at which two codebook vectors have the same value of distance function. Minimization of E may lead to a complicated treatment [97].

The set of values $\{\mathbf{m}_i\}$ that minimizes E is the solution of the VQ problem, and the signal space is mapped onto the set of codebook vectors. However, the indexing of these values can be made in an arbitrary way, whereby this mapping is still *unordered.*

One of the authors [97] has shown that if E is modified in such a way that the quantization error is *smoothed locally,* and the smoothing kernel h_{ci} is a particular type of function of the distance between units c and i in the array, then minimization of the new objective function

$$E' = \int \sum_i h_{ci} f[d(\mathbf{x}, \mathbf{m}_i)]p(\mathbf{x})d\mathbf{x} \quad (3)$$

defines *ordered* values of the $\mathbf{m}_i$, almost as if the $\mathbf{m}_i$ were lying at the nodes of an "elastic net" (see Fig. 1), which is fitted to $p(\mathbf{x})$ in the signal space. One may also state that orderliness of the $\mathbf{m}_i$ is a condition for the minimum of E' in many cases.

Unfortunately, the minimization of (3), even in the most fundamental cases, has not yet succeeded in closed form, and algorithms attempting to do it accurately have turned out to be too complicated for practical applications. Notwithstanding the so-called *stochastic approximation* [148] minimization of E' seems to yield fairly good approximations for $\{\mathbf{m}_i\}$ in general. Consider $\mathbf{x} = \mathbf{x}(t)$ at the discrete-time step t. Let $\mathbf{m}_i(t)$ be the approximation of $\mathbf{m}_i$ at time t, and consider the *sample function* $E''(t)$ of E', defined as

$$E''(t) = \sum_i h_{ci} f(d[\mathbf{x}(t), \mathbf{m}_i(t)]). \quad (4)$$

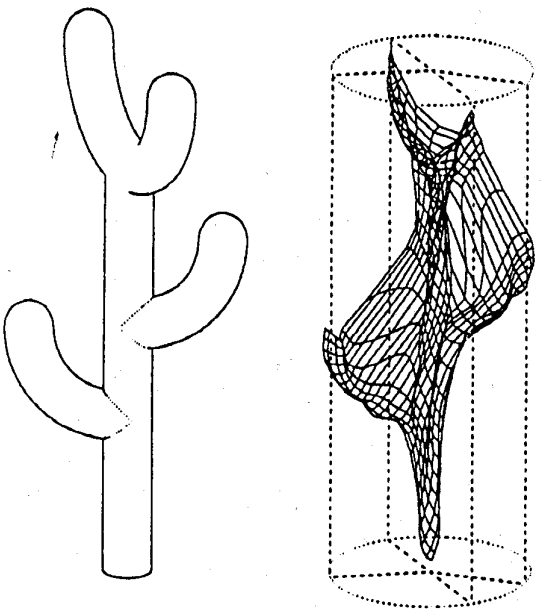


Fig. 1. An "elastic net" formed of the codebook vectors $\mathbf{m}_i$ (points connected by lines that describe the neighborhood relations) approximates here a three-dimensional (3-D), structured density function $p(\mathbf{x})$ (uniform within the "cactus," zero outside it) in an orderly fashion.

The approximate optimization algorithm is then

$$\mathbf{m}_i(t+1) = \mathbf{m}_i(t) - (1/2)\lambda(t)\partial E''/\partial \mathbf{m}_i(t) \quad (5)$$

where $\lambda(t)$ is a small positive scalar factor that determines the size of the gradient step at time t. This algorithm has been derived by approximating the gradient of E' at step t using the sample $\mathbf{x}(t)$ and approximate values $\mathbf{m}_i(t)$. This is a classical and thoroughly investigated approach [148]. If the function $\lambda(t)$ is chosen suitably, the sequence of the $\mathbf{m}_i(t)$ will always converge to the set of values $\{\mathbf{m}_i^*\}$ that approximates the solution for $\{\mathbf{m}_i\}$.

Many different SOM algorithms can be represented by (5). For instance, if $d(\mathbf{x}, \mathbf{m}_i) = \|\mathbf{x} - \mathbf{m}_i\|$ and $f(d) = d^2$, where the norm is Euclidean, we obtain the traditional SOM algorithm that was initially constructed with the "elastic net" analogy in mind

$$\mathbf{m}_i(t+1) = \mathbf{m}_i(t) + h_{ci}(t)[\mathbf{x}(t) - \mathbf{m}_i(t)]. \quad (6)$$

Here $\lambda(t)$ has been combined with $h_{ci}(t)$. The simplest but still very effective definition of $h_{ci}(t)$ is: $h_{ci}(t) = \alpha(t)$, where $\alpha(t)$ is another small scalar, if the distance between units i and c in the array is smaller than or equal to a specified limit (radius), and $h_{ci}(t) = 0$ otherwise. During the course of the ordering process, both $\alpha(t)$ and the radius decrease monotonically with time, whereby $0 < \alpha(t) < 1$. Let us call the set of neighboring units defined in this way $N_c = N_c(t)$.

We can express the simpler SOM equations as

$$\begin{aligned} \mathbf{m}_i(t+1) &= \mathbf{m}_i(t) + \alpha(t)[\mathbf{x}(t) - \mathbf{m}_i(t)] \\ &\qquad \text{if } i \in N_c(t), \\ \mathbf{m}_i(t+1) &= \mathbf{m}_i(t) \quad \text{if } i \notin N_c(t). \end{aligned} \quad (7)$$

The SOM algorithm can be further simplified in the following way. In the convergence limit every $\mathbf{m}_i = \mathbf{m}_i^*$ must satisfy the equilibrium condition

$$\mathrm{E}\{h_{ci}(\mathbf{x} - \mathbf{m}_i^*)\} = 0 \tag{8}$$

where E is the mathematical expectation value operator. An alternative way of writing (8) is

$$\mathbf{m}_i^* = \frac{\int h_{ci}\mathbf{x}p(\mathbf{x})d\mathbf{x}}{\int h_{ci}p(\mathbf{x})d\mathbf{x}}. \tag{9}$$

It has to be noted that $\mathbf{m}_i^*$ has thereby not been solved explicitly; the index c on the right side still depends on $\mathbf{x}$ and all the $\mathbf{m}_i^*$. Nonetheless, the $\mathbf{m}_i^*$ can be solved *iteratively*. For instance, if h_{ci} is defined as in (7), we have

$$\mathbf{m}_i^* = \frac{\int_{V_i} \mathbf{x}p(\mathbf{x})d\mathbf{x}}{\int_{V_i} p(\mathbf{x})d\mathbf{x}} \tag{10}$$

where V_i means the domain of vectors $\mathbf{x}$, whose nearest codebook vector belongs to the neighborhood set N_i of cell i. The iterative solution can then be expressed as the following steps. This algorithm, dubbed the *Batch Map*, resembles the familiar Linde–Buzo–Gray (LBG) algorithm for VQ [114], where all the training samples are assumed to be available when learning begins. The learning steps are defined as follows:

1) for the initial codebook vectors $\mathbf{m}_i$, take, for instance, copies of the first training samples $\mathbf{x}$;
2) for each map unit i, collect a list of copies of all those training samples $\mathbf{x}$, whose nearest codebook vector belongs to the neighborhood set N_i of cell i;
3) take for the new value of each new codebook vector the mean over the respective list;
4) repeat from step 2) a few times.

C. Guidelines for Reading

Many tutorial texts and surveys on the SOM have been published [85], [102], [93], [145]. A software package that contains all the central procedures, a number of monitoring and diagnostic programs, and exemplary data is freely available in the Internet [99]. Over 2300 research papers have been published on the SOM; a list of them is available in the Internet [99].

In view of this affluent information available on the SOM, we are not going to describe in this paper any more refined details of the algorithm itself; if the readers intend to use the SOM, they are urged to consult the original references. Our purpose here is to concentrate on considering, with the aid of a few concrete case examples, how to choose the variables for a particular problem, how to preprocess them, how to dimension the map, and how to combine the SOM with other algorithms.

Section II describes general principles of preprocessing. Sections III–VI discuss selected cases of the application of the SOM to some major engineering fields, and Sections VII and VIII will complete the discussion.

II. Preprocessing and Feature Extraction

Only in few particular applications of the SOM, e.g., exploratory data analysis, the primary data may be used almost directly as such. In most engineering applications the neural networks constitute the interface between real-world situations and machines, whereby the measurements may be made using very sophisticated instruments. The real-world measurements contain variations for very different reasons: due to movements of the objects, differing illuminating conditions, heavy disturbances and noise, aging of the devices and their faults, etc., which then should be eliminated or compensated for. Especially in computer vision, it is often necessary to extract a set of *invariant features* from the primary observations, and this is frequently done by nonneural preprocessing methods. The share of preprocessing of all computations may be much larger than that of decision making.

The purpose of Section II is to emphasize the importance of preprocessing and to give a few general practical advices.

A. Selection of Input Variables and Their Scaling

Consider first a set of simple measurements, whereby invariances of signal patterns are not yet taken into account. The measurements may have been taken with very different devices and in different scales. In statistical decision making, the set of measurements is often regarded as a *metric vector* that may be analyzed relating to the Euclidean space. The various elements of the vectors should therefore represent information in as balanced way as possible.

If the number of measuring channels or variables to be considered is very large, i.e., thousands, it will be necessary to select a much smaller number of representative variables to the inputs of the neural network. To this end one might analyze the information-theoretic entropy measure of the different channels to decide which variables should be selected.

To equalize the contributions of the different variables to the classification results, the simplest method is to *normalize* all variables into scales in which, e.g., the mean of each variable is zero and its variance is the same.

In the general case the variables are mutually dependent, i.e., their joint distribution is structured and cannot be expressed as the product of marginal distributions. In such a case it is advisable to first make a clustering analysis of the measurements, e.g., by the *Sammon projection method* [150] in order to deduce whether simple rescaling is sufficient to separate the clusters, or whether the primary measurements should be somehow combined to eliminate their interdependence.

In Section III we will discuss two particular examples that are characteristic of measurements made in engineering, and for which a simple rescaling has been sufficient.

B. Feature Extraction

The real-world signals are usually transformed into measurements in their passage through different systems, often characterizable by some integral transforms, and the ba-

sic form of the latter is *convolution*. The convolution kernel is called the instrument function (in measuring techniques), the point spread function (in vision), or the impulse response (in signal theory). It is often advisable to try to improve the detection and classification of source information using suitable inverse transforms or their approximations.

Sometimes, however, intentional *fuzzification* or *blurring* of input patterns is useful, for instance, to improve the tolerance of their detection to shifts in the input field.

A typical task in engineering is to identify various distinct *objects* from their observations. As they may appear in different variations, direct identification of their images on the basis of the exact constellations of their pattern elements is usually not advisable. One principle, for instance in the recognition of faces or written characters well-positioned in the input field, is to describe the input data in terms of some *eigenvectors*. For instance, if the vector formed of all pixels of an image is regarded as a real vector $\mathbf{x} \in \Re^n$, then one might first compute the correlation matrix

$$\mathbf{C_{xx}} = \frac{1}{N} \sum_{t \in S} \mathbf{x}(t)\mathbf{x}^T(t) \tag{11}$$

using a large set S of sample images (here N is the number of elements in S). If the eigenvectors of $\mathbf{C_{xx}}$ are denoted $\mathbf{u}_k \in \Re^n, k = 1, 2, \cdots, n$, then any image vector $\mathbf{x}'$ can be expanded in terms of the eigenvectors,

$$\mathbf{x}' = \sum_{k=1}^{p} (\mathbf{u}_k^T \mathbf{x}')\mathbf{u}_k + \varepsilon \tag{12}$$

where $p \leq n$ and ε is a residual. A restricted set of the largest factors $\mathbf{u}_k^T \mathbf{x}'$ called the *principal components* can be used in (12) to describe the approximate appearance of $\mathbf{x}'$ [133]. This is an example of a *feature set* that is a compressed description of $\mathbf{x}'$ at some statistical accuracy. There also exist numerous other eigenfunctions of the input data, in terms of which the feature set can be defined.

When using features that refer to the whole input field, the images have to be *standardized*. Frequently, especially in the recognition of moving targets, this is not possible, however. The signals emitted by the object are transformed into observations in a complex way. In biology as well as artificial perception the solution is then to extract from the primary observations a set of *local features* that are more or less *invariant* with respect to the natural transformations.

For instance, in image and signal analysis, the so-called *wavelets* have recently gained much importance as such local features. The two-dimensional (2-D) *self-similar wavelet* or the *Gabor function* [45] is defined over the image field as the complex function

$$\psi_{\vec{k}}(\vec{x}) = \exp[i(\vec{k} \cdot \vec{x}) - (\vec{k} \cdot \vec{k})(\vec{x} \cdot \vec{x})/2\sigma^2] \tag{13}$$

where $\vec{x}$ is the location vector on the image plane, and $\vec{k}$ is the wave vector, respectively. The wavelet transform F of function $f(\vec{x})$ is

$$F = F(\vec{k}, \vec{x}_0) = \int \psi_{\vec{k}}(\vec{x} - \vec{x}_0) f(\vec{x}) d\vec{x}. \tag{14}$$

Its modulus $|F(\vec{k}, \vec{x}_0)|$ is almost invariant with respect to displacements of $f(\vec{x})$ around location $\vec{x}_0$ in a domain where the radius is of the order of σ. To the neural-network inputs one can select a set of such local features $|F(\vec{k}, \vec{x}_0)|$ with different values of $\vec{k}$ and $\vec{x}_0$. Referring to the same $\vec{x}_0$, such a set is also called the *Gabor jet* [15].

In the recognition of acoustic signals such as speech, local-feature extraction can be made in a computationally effective way by taking fast discrete Fourier transforms over specified time windows. If the signal is generated as a set of acoustic resonances driven by pulsed excitation, like the speech signal is, then the driving function can be eliminated by taking two successive Fourier amplitude transforms of the signal and using a mask for the driving function. The result of the first transform is usually logarithmized: the feature set thereby computed is called the *cepstrum* [33].

In this paper we shall also discuss the analysis of textural information in images. There exist numerous texture models and choices for textural features. In connection with the SOM we have found the so-called *co-occurrence matrix* [57] useful; its elements, and in particular statistical measures of the elements, can be used for the characterization of texture.

C. Emergence of Invariant-Feature Detectors in the Adaptive-Subspace SOM

In practical applications, the analytical forms of the translation-invariant local-feature detectors such as Gabor filters or other wavelet transforms have been selected heuristically. It is a question of great theoretical importance, however, whether such detectors could *emerge* in adaptive processes, in response to real signal statistics, without any prior assumption of the kernel forms.

One of the authors has recently [102]–[104] introduced a new algorithm, called the *Adaptive-Subspace SOM* (ASSOM), that has been found to create such filters automatically. These filters are produced by their adaptation to real data. However, it is not in the first place the form of the input patterns, but the way in which the patterns are presented to the learning system, that defines the filters. Therefore almost the same kind of filters are formed in response to artificial noise, photographic images, etc.

The ASSOM is a *subspace classifier* [102], [181], [84] in which the basic operational unit is not a single neuron but a two-layer group of neurons. The input weights of the first-layer neurons correspond to orthonormalized *basis vectors* of the due subspace, and the input layer computes the dot products of the input vector with the basis vectors. A neuron in the second layer computes a quadratic function of the first-layer outputs. The output from this module then describes the *length of projection of* $\mathbf{x}$ *on the subspace* $\mathcal{L}_i$ *spanned by its basis vectors.*

In the ASSOM, an array of such units, each describable by its own subspace $\mathcal{L}_i$, forms the SOM array. Matching of $\mathbf{x}$ is made by computing the projections of $\mathbf{x}$, denoted $\hat{\mathbf{x}}_i$, on each subspace, and defining the winner unit c as

$$c = \arg\max_i \{\|\hat{\mathbf{x}}_i\|\}. \tag{15}$$

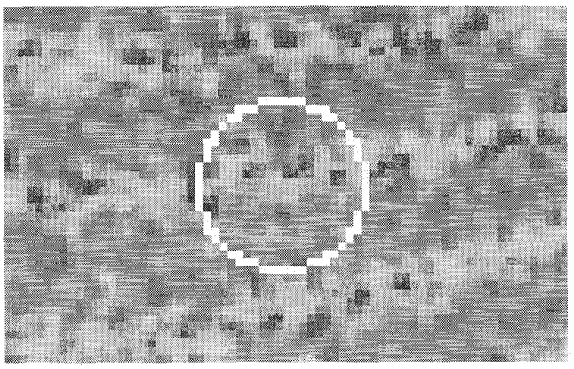
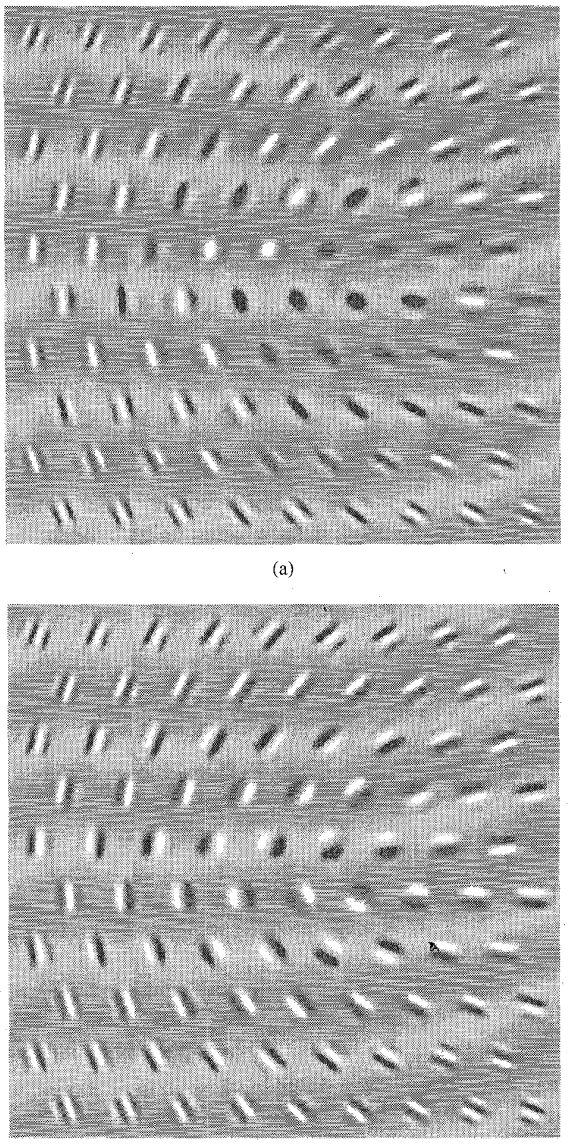


Fig. 2. The images in this experiment consisted of low-pass filtered noise, moving with respect to the receptive field (white circle).

The adaptive process of the ASSOM is different from that taking place in the traditional SOM. Instead of the weight vector of the winner c, the weight vectors of all the basis vectors of the winner subspace (and subspaces in its neighborhood N_c) are rotated simultaneously in proper directions such that $\|\hat{\mathbf{x}}_i\|, i \in N_c$ is increased.

When using the ASSOM to learn pattern transformations that form various signal subspaces, a whole *episode* of input patterns is shown as one operation; then the transformation implicit in this episode will be learned by the units of the ASSOM around the winner. Consider Fig. 2, where the image field contains low-pass filtered noise, and the white circle represents the receptive field with 316 pixels in it. An episode can be formed by moving the image relative to the receptive field. In the adaptive learning process, the basis vectors of the different ASSOM units converge to values that resemble the Gabor functions. For different transformations of the image during the episode, the filters converge to different patterns. In Fig. 3 we show the result for an ASSOM in which each subspace was spanned by exactly two basis vectors, and the forms of the basis vectors emerged corresponding to the cosine- and sine-type Gabor function, respectively.

Above the ASSOM algorithm was only delineated rather superficially. Its theory and details are presented in [102]–[104].



(a)

(b)

Fig. 3. The SOM was a 9 by 10 array of units, each having two 316-element basis vectors, their format corresponding to the receptive field in Fig. 2. Here the basis vectors are shown in two separate pictures, relating to the same SOM units, in a gray scale.

III. Process and Systems Analysis

An industrial plant or complex machine ought to be described in terms of state variables, the number of which may exceed the number of measurements by an order of magnitude. The state variables may also be related in a highly nonlinear way. In the sense of classical systems theory, the analytical model of the plant or machine would then not be identifiable from measurements. Nonetheless there may exist much fewer characteristic states or state clusters in the system that determine its general behavior and are somehow reflected in the measurements. As the SOM is a nonlinear projection method, such characteristic states or clusters can often be made visible in the self-organized map, without explicit modeling of the system.

An important application of the SOM is in fault diagnosis. The SOM can be used in two ways: to detect the fault and to identify it. In practical engineering applications, we can distinguish two different situations; either we have no prior measurements of the faulty situations, or we have also been able to record the faults.

This section first describes two cases that illustrate the visualization of machine states and identification of fault situations. The applicability of the SOM to continuous processes is then discussed from a general point of view.

A. Visualization of Machine States

Understanding and modeling of complex relationships between variables in large systems is often problematic.

Automated measurements produce masses of data that may be very hard or even impossible to interpret. In many practical situations, even minor knowledge about the characteristic properties of the system would be helpful. Therefore, the online measurements ought to be converted into some simple and easily comprehensible display which, despite the dimensionality reduction, would preserve the relationships between the system states. With this kind of transformation available, 1) it would be possible for the operating personnel to visually follow the development of the system state, 2) data understanding would facilitate estimation of the future behavior of the system, 3) abnormalities in the present or predicted behavior of the process would make identification of fault situations possible, and also 4) the control of the system could be based on the state analysis [158].

The SOM represents the most important structures of the density function of input data in a low-dimensional display. Together with proper visualization methods the SOM is thus a powerful tool for discovering and visualizing general structures of the state space.

Therefore the SOM is an efficient tool for visualizing the system behavior, too. Consider a machine from which several measurements relating to the system itself as well as to its environment are taken. Denote the measurement or feature vector by $\mathbf{x} = [\xi_1, \xi_2, \cdots, \xi_n]^T$. In addition to the system variables, control variables may be included in the feature vector. The scale of each measurement is normalized so that either the maximum and minimum of each ξ_j, respectively, are equal, or the variance of every ξ_j is the same.

Assume that enough measurements are available for the SOM algorithm that forms the weight vectors $\mathbf{m}_i = [\mu_{i1}, \mu_{i2}, \cdots, \mu_{in}]^T$. The map array so created is then fixed prior to its use in monitoring.

Any component plane j of the SOM, i.e., the array of scalar values μ_{ij} representing the jth components of the weight vectors $\mathbf{m}_i$ and having the same format as the SOM array, can be displayed separately, and the values of the μ_{ij} can be represented on it using shades of gray.

The SOM can then be used in two ways: 1) the *trajectory* consisting of successive images of operating points can be displayed *on the original SOM*, on which some labels may indicate general information about the system states, or 2) the same trajectory can also be drawn on the display of any of the *component planes*, whereby the shades of gray along the trajectory directly indicate the values of that process variable over time.

A practical example is shown in Figs. 4 and 5, where a power transformer has been analyzed [78]. In this example, the states relating to the measurements of ten variables have been followed during a period of 24 h.

In Fig. 4, the trajectory has been drawn on the component plane that displays the load current of the transformer. Dark gray tones correspond to small and light tones to heavy loads, respectively. It can be seen that the operating point has moved from dark to light areas and back again corresponding to the daytime operation of the system, when the load was switched on and off, respectively. In this specific example, the trajectories of the successive days were very similar during normal operation. The operating personnel can easily follow up the complete system behavior.

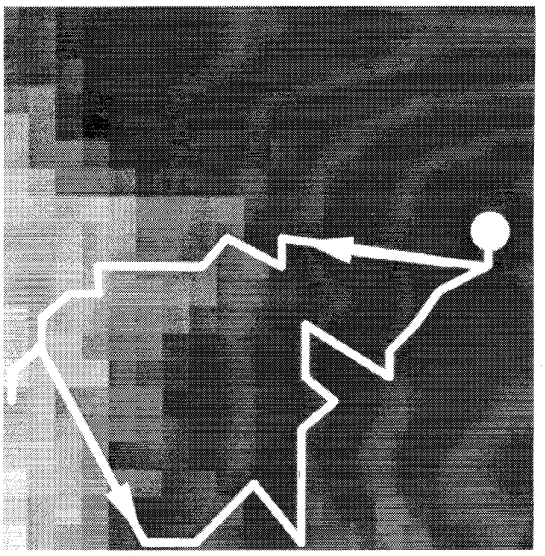


Fig. 4. The square array shown in this illustration has the same format as the SOM array, but each small square in it only represents the value of one component j of some weight vector $\mathbf{m}_i$, in a gray scale. The trajectory drawn in white describes the sequence of "winner" units on the SOM, in response to a sequence of input vectors $\{\mathbf{x}(t)\}$ taken over some period of time. This picture describes the load current (white: high, black: low), whereby its values can be read as a function of time from the gray-shade values along the trajectory. The form of the trajectory is defined by the sequence of states of the transformer, as a function of switching the load on and off, during 24 h of operation. In normal conditions, the trajectories look very similar on successive days.

In Fig. 5, the component planes of eight parameters of interest have been picked up for display. The same trajectory of the operating point on the SOM has been drawn on each separate display indicating the respective parameter values during operation.

B. Fault Identification

During training, the weight vectors of the SOM become adapted to that domain of the system state space from which the measurements have been taken. Thus the state space will be divided into two parts: 1) The operation space represented by the SOM and 2) its complementary space. In the previous example, only the situations including the normal training data were identified by the SOM, whereas any fault situation would drive the trajectory somewhere to the edges.

Fault detection can also be based on the *quantization error*: when the feature vector corresponding to the measurements is compared with the weight vectors of all map units, and the smallest difference exceeds a predetermined threshold, the process is probably in a fault situation. This conclusion is based on the assumption that a large quantization error corresponds to the operation point belonging to the complementary space not covered by the training data.

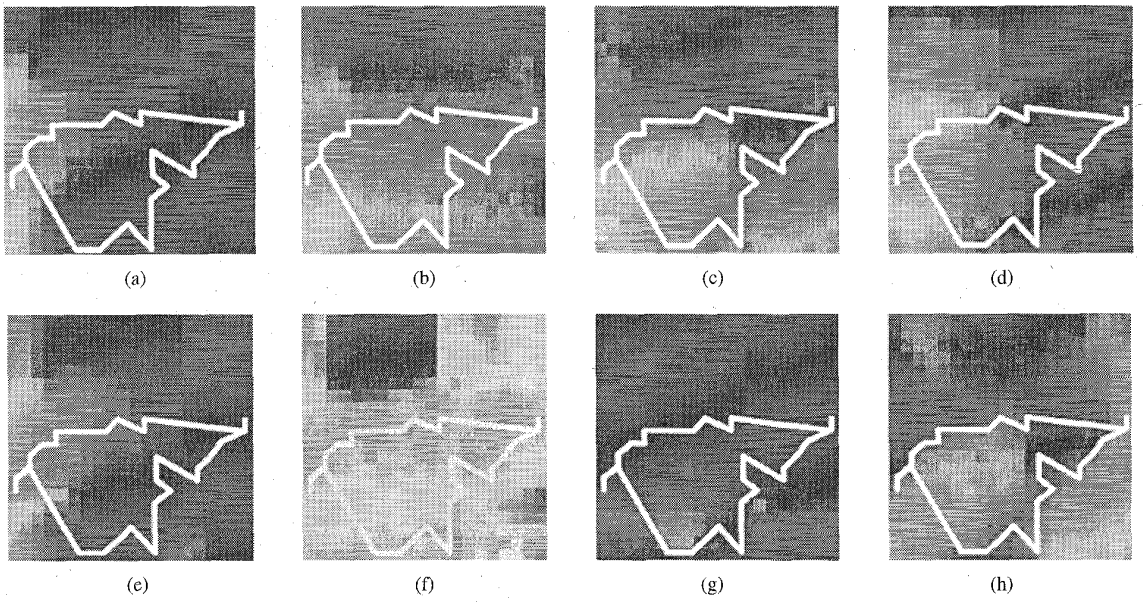


Fig. 5. Eight component planes of the weight vectors $\mathbf{m}_i$, representing the following variables in respective gray scales, are shown: (a) Load current, (b) coil temperature, (c) average oil temperature, (d) oil temperature on top of the transformer, (e) position of tap changer (controlling the load), (f) voltage, (g) hydrogen content, and (h) temperature of transformer room. The same trajectory has been superimposed on each of these planes to facilitate the reading of the different measurements as a function of time. Visual inspection of the component planes also shows clear correlation between the various parameters.

Therefore, the situation is new and something is possibly going wrong.

In addition to visualizing the normal operating conditions by the SOM, however, it would be desirable to be able to visualize fault states, too. The main problem thereby is where to get abnormal but still typical measurements from. The faults may be rare and true measurements thus not available. In some cases, fault situations can be produced during testing, but especially in the case of industrial processes and big machines, production of severe faults might be too costly. Then the faults must be simulated.

If fault situations and their reasons are known well enough, simulated data for SOM training can be produced easily. In practical engineering systems, the different types of faults or errors most usually encountered are: 1) a break in signal lines resulting in abrupt drop of a signal, 2) heavy disturbance with a large sporadic change in signals, 3) jamming of a measurement value, often caused by a mechanical fault in measuring equipment, 4) slow drift in measured value due to aging of the device, and 5) a sudden change in some regulated parameter value, indicating a fault in the control mechanism. All the mentioned fault types can be simulated easily and independently. Thus the feature vector x used in training can be varied artificially (by causing mechanical and electric disturbances) or by simulation, and "danger" areas can be created on the SOM.

If measurements of most of the typical fault types are available or can be simulated, mechanically or by modeling, the SOM can be used as a monitor of the operating conditions of the system. Fig. 6(b) describes the SOM computed for an *anaesthesia machine*, and different areas on it correspond to obstruction of tubes, hoses, or pumps, their breakage, wrong gas mixtures, positioning errors of the intubation tube, etc.

In systems where the operating point is not stable, or where it is not possible to define "normal" situations such as a "normal" patient in this example, the SOM must be used on two levels for the analysis of faults or alarms. On the first level, the so-called *fault-detection map* is used. The fault detection is based on the *quantization error*. On the second level, a more detailed map (or an "atlas" of maps) is used to identify the reason of the fault by tracing the trajectory of the operation point. By storing a sequence of feature vectors, the behavior of the process before and during the occurence of the fault can be analyzed in more detail. Examples of fault detection and identification in an anaesthesia system are described in [174]. The complete *anaesthesia system* comprises the anaesthesia machine, the patient, and the anaesthesia personnel. The purpose of the SOM-based monitoring system is to minimize the risks of anaesthesia by detecting and identifying the faults before they cause injury to the patient. Fig. 6 shows detection of the fault based on the quantization error and identification of the fault from the trajectory of the operating point.

C. Process Analysis and Monitoring

There exists a wide variety of industrial processes. It may suffice here to express some general views about a typical process, such as a continuous-flow chemical process, referring to general problems in its monitoring and control. It is characteristic of chemical processes that their time

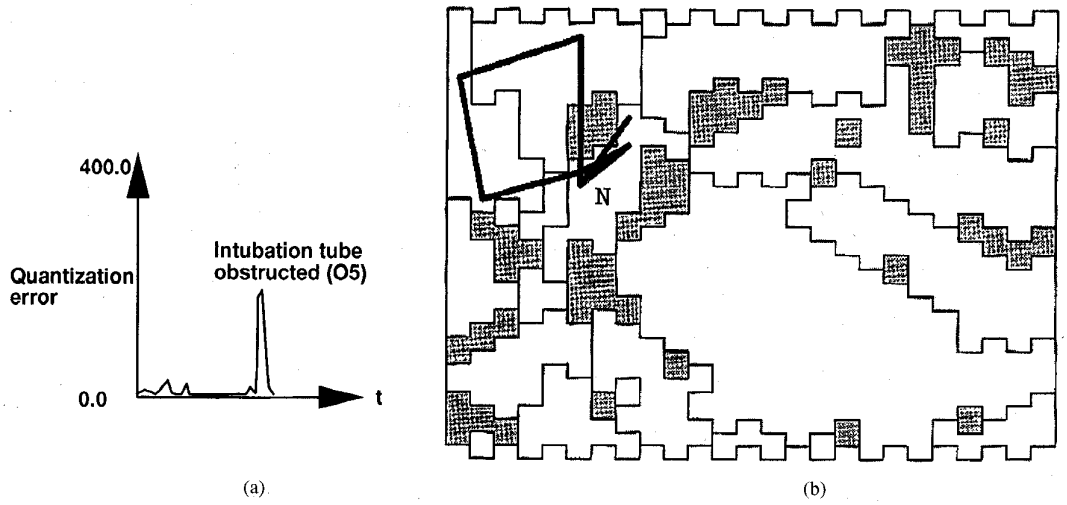


Fig. 6. Fault identification of an anaesthesia system tested in a true situation: N = normal state.The position of the patient was changed and the intubation tube was obstructed for a short period of time (fault 05). The increase in the quantization error shown in (a) indicates that a fault has been detected. The trajectory of the operating point was moving from the area corresponding to the normal situation to the area that corresponded to an obstruction in the specific part of the system. The trajectory is depicted in (b).

constants are generally very large, whereas most control actions affect the process states in a nonlinear way.

Traditionally, in order to be able to monitor and control the process, a global system model ought to be available, and it must be possible to place a sufficient number of transducers to places where they measure essential process variables. Typical problems are thus that the most relevant places in the process are frequently not accessible, and the most essential properties of the product may only be measurable off-line. Therefore, estimation of process or signal states must frequently be made indirectly. For instance, elaborate methods such as multivariable analyzes or computation of cross correlations and power spectra have to be used; long series of measurements are thereby needed, and still the statistical dependencies may only be obtainable in the linear approximation.

The SOM is a new method for the analysis of the process states and their stability. When using it, a physically or chemically definable process model is not necessary. The SOM is a statistical method, too, but it has the desirable property of describing *nonlinear* relationships between a great many different variables *phenomenologically*. The map is computed from normalized measurements over a training period, as exemplified in Sections III-A–B (and the map can also adapt to new measurements continuously). When the display, describing all occurring situations, is labeled by calibration measurements, it will be possible to monitor the process state by following the trajectory of the operating point on the map. Under normal operating conditions such as those described in Section III-A, the process variables may be regarded to be in a dynamic equilibrium, whereby the SOM produces a rather faithful image of the process state. This is no longer true if a fault causes a transient situation, whereby due to different delays the state variables need some time to settle down. In the case discussed in Section III-B the time constants of the machine were still of the same order of magnitude as the length of the error situation. In typical industrial processes, however, abnormalities in the trajectories usually only indicate that something is going wrong. The set of measurements does not necessarily identify the process state uniquely.

The SOM, however, normally facilitates direct control of the process: for instance, the operating personnel may learn to adjust the control variables in such a way that the trajectory stays in the allowed region, and locally linearized models for that may be set up easily; the correct control action may also be learned adaptively, as described in Section V.

As the SOM can cover many degrees of freedom of the process, it may also be possible to deduce from an abnormal trajectory whether there exist faults such as damages or contamination of the transducers.

D. Survey of Applications of the SOM to Process Analysis

This section involves various process analysis applications of the SOM, excluding the image analysis and robotics research that are reported later in Sections IV and V.

In an attempt to take the process dynamics into account, the SOM algorithm may be slightly modified [70]. Other system-theoretic aspects added to the SOM analysis have been described in [8], [188], [189], [27], [34], [118], [169], and [187]. Large-scale diagnostic problems are discussed in [119]. Identification of the process state, in addition to the works already described above in Sections III-A–B, is described in [51], [52], [132], [160], [171], and [172]. Further works on the following problems are: fault diagnosis [36], [161], [163], diagnosis of machine vibrations [185], detection of diagnostic symptoms of the plant [44] and process error detection [3], [4].

A major application area is power engineering, where the SOM has been used to analyze the following tasks: electricity consumption [122], consumption alarms [9], load forecasting [69], [113], time of occurrence of electric utility peak loads [126], power flow classification [129], system stability [120], [121], power system static security assessment [37], [127], [128], [130], [131], impulse test fault diagnosis on power transformers [12], partial discharge diagnosis [152], [153], and avoidance of the high-voltage mode of electromagnetic voltage transformer [192].

Miscellaneous process applications are: monitoring of paper quality [108], analysis of breaks in a paper machine [162], classification of wooden boards [191], timber classification [31], and operation guidance of a blast furnace [105].

IV. Statistical Pattern Recognition

The classical theories of pattern recognition and image analysis have been developed over dozens of years. However, with the advent of artificial neural networks, *nonlinear statistical models* for natural sensory data, with a very high dimensionality, became computable.

Systems that involve neural networks in their most computation-intense parts are usually hybrids of traditional and neural methods. The main purpose of this section is to exemplify how the SOM algorithm can be combined with the more traditional methods.

A. Computer Vision

Computer vision (or machine vision) is one of the potential application areas of neural networks. It has an important role in industrial automation such as visual quality control and robotics, as well as in document processing, medical imaging, remote sensing, and target recognition. All but the most trivial problems require a number of processing steps, by which the digitized input images are preprocessed and segmented into relevant parts, features are extracted to represent the segmented regions or objects, and the feature representations are classified and interpreted. The predominant approach in feature extraction and classification has been model-based; *a priori* knowledge about the objects is programmed as a set of relations, and artificial intelligence-type inference is used for interpretation [10].

Recently, this approach has been criticized due to its inflexibility: there are many object classes such as handwritten characters, human faces, natural textures, and surface faults, which are very hard to model explicitly because of the inherent nonrigid variations that naturally occur. Learning methods that form some kind of internal experimental models based on training data are a promising alternative. However, then the problem may be the availability of statistically sufficient amounts of *preclassified* training data; typically the neural networks suggested for computer vision problems are very large, having a large number of parameters and thus requiring very large training sets.

Using the SOM as a feature extraction method in computer vision is a promising approach, because the learning is *unsupervised*; no preclassified image data are needed at all. When highly compressed representations of images or their parts are formed by the SOM, the final classification or labeling stage can be fairly simple, needing only a moderate number of labeled training samples. In some low-level tasks like learning visual detector masks, image windows as such can be used as inputs to the SOM. In most cases, however, some prior feature extraction has to be performed. Each specific problem and image data type has its own characteristic features which have been experimentally found to give good representations, for instance wavelet filters, edge detectors, texture measures based on gray level co-occurrences, color, or multispectral information, etc. Thus domain knowledge about the problem can and should be used at this first level, prior to using a learning neural network. Also typical invariances occurring in the images should be handled by the feature extraction stage. A challenging task is whether also the low-level image filters can be automatically found from raw data; a solution is given by the ASSOM network described in Section II-C.

An example of face classification based on [109] is described here to illustrate a typical SOM approach. A similar approach is directly applicable, for instance, to the analysis of knots in boards and local faults in other materials.

These are the main consequent blocks of the system:

1) a fixed preprocessing stage, in which Gabor filters (13) of varying orientation and resolution are applied to each image location;
2) a map layer, using SOM networks, that takes the output vectors of the Gabor filter bank as inputs and clusters the Gabor vectors into a set of characteristic codes;
3) a supervised classification stage, whose inputs are histograms of the individual images.

Fig. 7 shows the feature extraction and compression blocks. Using the codes thus obtained, it is possible to represent an image as a histogram that gives for each code the number of image regions that were mapped to that code value. These histograms are able to discriminate between different items and tolerate moderate variations of scale, orientation, and nonrigid transformations (like changes of facial expression in this example). In this concrete experiment, images of a set of 17 persons taken from different orientations were correctly classified although only one training image for each person was used to construct the code histogram [109], [135].

B. Texture Analysis and Classification

Texture analysis is an important part of image analysis in general, for instance in the segmentation of meaningful areas from the images. It may also be directly applied in industry to the visual monitoring of the quality of materials like paper or wood surfaces.

The meaning of the term "texture," however, is vague. Intuitively, it is easy to give examples of textures such as

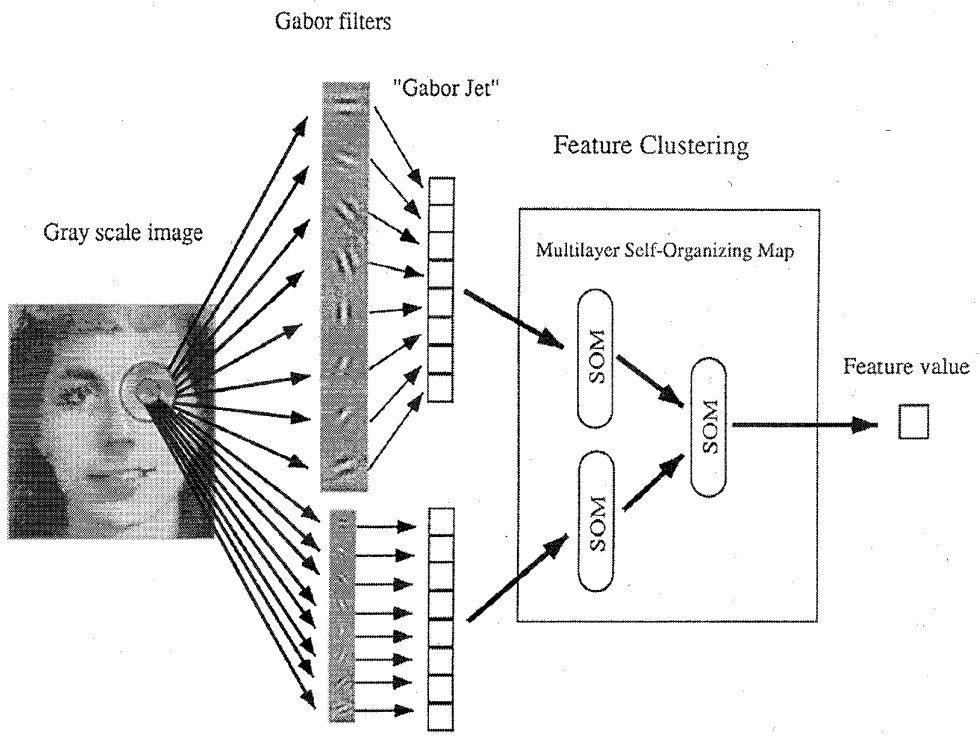


Fig. 7. Schematic drawing of feature extraction and compression in object recognition. First, a bank of 16 Gabor filters is applied to all image windows. Gabor filters are orientation- and spatial-frequency-selective masks that detect characteristic microfeatures of the face images. Each small region in the image is represented as a "Gabor jet" (set of outputs from the 16 local Gabor filters). These vectors are clustered by a two-layer SOM that has been trained by a number of face images and other natural objects. The first-layer SOM's divide their input space into convex clusters centered around the map weight vectors, and the second SOM combines these clusters into 100 nonconvex clusters that are able to capture small variations in the images. The cluster number is used as a feature code for that image region.

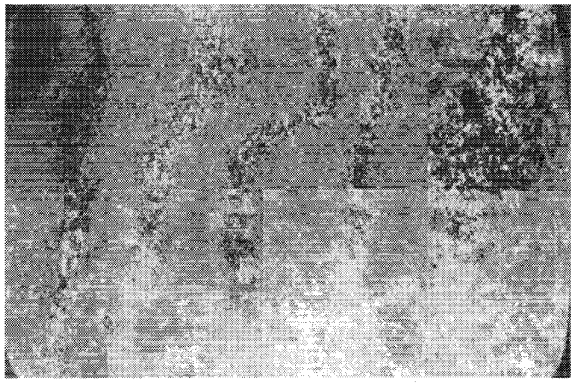


Fig. 8. Some characteristic stochastic textures found in paper. These kinds of textures are discussed in the present paper.

woven materials, brick walls, grass, fur, etc., but defining the term is difficult. VanGool [173], however, has succeeded in expressing the following that may be accepted for the definition of the texture in general: "Texture can be defined as a structure composed of many more or less ordered similar elements or patterns, without any of them drawing special attention." Examples of eight textures occurring in paper are given in Fig. 8.

In this article we only consider texture analysis from the engineering point of view. The principal phases in texture analysis are data acquisition, feature selection, clustering, and classification. The approach described in this Section is illustrated in Fig. 9. After extracting a set of suitable features from image windows, the feature vectors are clustered and classified by the SOM directly. The produced set of textural prototypes, the weight vectors of the SOM, are calibrated by known samples. The map is then called the *texture map* [176]. If necessary, a certain amount of fine tuning of the weight vectors is made by the *Learning Vector Quantization* (LVQ) algorithm [87]. The texture map so defined can be directly used for classification. The map inherits from the SOM the property that adjacent neurons resemble each other, which is important in the reliable comparison of textures.

Image acquisition and preprocessing is a particularly important phase of successful texture classification. Not only are the dimensions of the image windows and the spatial resolution essential, but data acquisition should be as accurate and repeatable as possible, especially if photographic phases are involved. This arrangement should remain unchanged during the training and classification processes. Variations in illumination, either by visual light

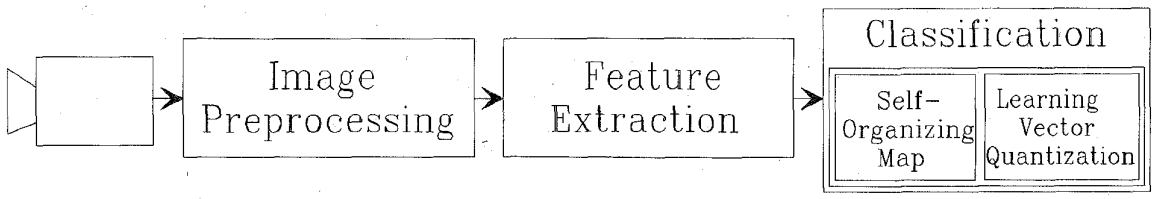


Fig. 9. Block scheme of texture classification.

or other sources of radiation, must be compensated for. For instance, one may normalize the intensity and contrast by equalizing the intensity histograms [28].

In a practical application such as paper formation, expert knowledge in the selection of textural features is still needed. Several standard approaches to it have been developed [58], [173], [59], [28]. The texture descriptors should involve as few limiting assumptions as possible. This usually means that only statistical descriptors are valid. A significant amount of statistical information about texture is already contained in the statistics of pixel pairs. This is usually represented by the so-called *co-occurrence matrix* $\mathbf{M}_d = \mathbf{M}_d(i,j)$ [56], [183], [29]. Let the scalar values $s(i), i = 1, 2, \cdots L$ define an L-level gray scale. The element (i,j) of matrix $\mathbf{M}_d$ expresses the number of pixel pairs in the image window with relative displacement d, the first pixel having the value $s(i)$ and the second $s(j)$, respectively. For textural features one can then take representative samples of the elements of $\mathbf{M}_d$ directly, or some statistics of them [56].

A typical problem in classification is that the shapes of the feature distributions are unsymmetric and even very oblongated. Traditional parametric classifiers thereby do not work well [167]. Neural network classifiers are superior due to their ability to learn arbitrary decision surfaces from examples.

By the current techniques, it is possible to acquire large amounts of raw measurements, especially if their classification need not be defined. Among neural network methods, the SOM is the proper tool to study the data dependencies in large amounts of unlabeled data. The unsupervised learning rule of the SOM is not very sensitive to the quality of the data; it is, however, important that all typical classes are well represented. The amount of data should be large enough to allow the estimation of the mixture probability density of the feature vectors. In practice, this means hundreds of samples per a required class.

Classification based on the SOM is already possible without any supervised learning. The classification accuracy, however, increases significantly (two- or three-fold) if the map vectors are tuned in a supervised way. For this, known samples are needed. Calibration or labeling of the map units for them to correspond to texture classes is usually made by majority voting [175]. It is usually required that at least some tens of known samples are available in each class. The quality of these samples is very important; they should characterize each class well.

After labeling, the overall classification rate is computed. Usually only some of the classes need to be fine tuned by LVQ. There exist several versions of LVQ [87], [89], [92].

With stochastic textures, it may be a problem to obtain correctly classified reference samples. Usually, a human expert team is needed, and the experts make majority decisions. The error rate is estimated by mixing the samples and classifying them again several times. The automatic analysis method should be robust enough to tolerate the residual human errors. For instance, a very common error, due to nonhomogeneity of a sample, can be compensated for with a large map size. It is possible to reserve a neuron in the texture map to this nonhomogeneous type of texture.

C. Speech Recognition

Speech recognition is not an engineering application in itself, but it can be used, for example, to give commands to machines. One reason for describing it here is that similar methods may occur in the analysis of other acoustic signals, too, for instance, in the diagnosis of engine conditions on the basis of sounds from the bearings or acoustic emission indicating deformation of materials.

The exemplary problem discussed here is to develop a speech recognition system that transcribes unlimited speech into orthographically correct text, a kind of "dictation machine." The system is speaker-dependent, i.e., speakers must have their own speech models in the recognition system. The system recognizes phonemes and constructs the final text of them. Such a recognizer was developed for the Finnish language.

The complete speech recognition system is a hybrid of four subsystems (see Fig. 10). First, the speech signal is preprocessed in order to extract spectral features (cepstral coefficients) from it. The features are classified in the second stage into a *quasiphoneme* string every 10 ms. In the third stage, the sequence of quasiphonemes is segmented, each segment being supposed to represent one *phoneme* of speech. In the postprocessing stage, the phonemes are collected into words or phrases, and grammatical methods are used to correct systematic errors in symbol strings caused by coarticulation effects.

Spectral analysis of the speech signal has been a standard method in speech preprocessing. The two main alternatives for feature sets are the *short-time Fourier amplitude spectra* as such, and the nonlinear combination of two cascaded Fourier transformations, which results in a feature set called the *cepstrum*. In literature the cepstrum has been proposed as a very compact representation of the speech signal [33],

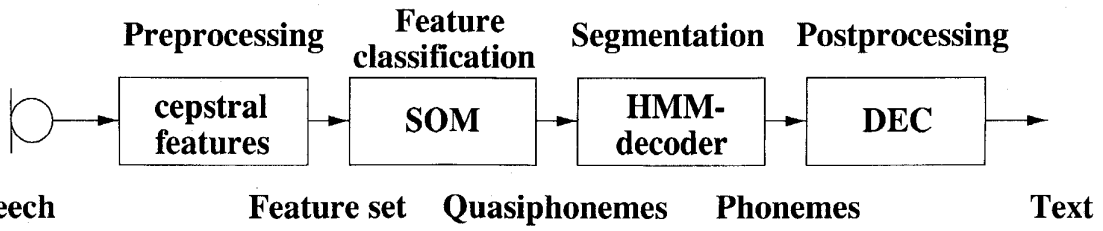


Fig. 10. Configuration of the speech recognition system. In the first stage the speech signal is preprocessed to extract spectral features (cepstra) from it. In the second stage the features are classified into a quasiphoneme string. In the third stage, the sequence of quasiphonemes is segmented, each segment representing one phoneme of speech. In the postprocessing stage, the phonemes are collected into words, and grammatical methods are used to correct systematic errors in symbol strings caused by coarticulation effects.

and one of the main reasons for using it is the possibility of eliminating the fundamental (glottal) frequency of speech by rejecting the corresponding band from the cepstra.

We have used cepstra in our current speech recognition systems. Preprocessing of the speech signal contains the following steps [168]: The speech signal is collected using a sampling rate of 12.8 kHz. A 128-component logarithmic power spectrum is computed every 10 ms, using 256 successive speech samples emphasized by a Hamming window. The 128-component spectrum is compressed into 22 spectral components X_k by combining frequency channels into fewer components in the so-called mel-scale (ten linearly spaced components below 1 kHz, 12 exponentially spaced components between 1 kHz and 5 kHz). The cepstral coefficients C_i (20 components) are computed by a discrete cosine transform of the filtered power spectra as

$$C_i = \sum_{k=1}^{N} X_k \cos\left[i\left(k - \frac{1}{2}\right)\frac{\pi}{N}\right], \quad i = 1, 2, \cdots, M \tag{16}$$

where N is the number of spectral channels and M is the number of cepstral coefficients.

Recognition accuracy can be improved by concatenating several successive cepstral vectors in a time window into a higher-dimensional feature vector which, therefore, also contains information about the temporal relations in the signal [74]. We have used three- to seven-fold concatenation.

The basic SOM algorithm is *unsupervised*, and it basically describes the degree of clustering of the input data. However, when the SOM was originally suggested for classification tasks, it turned out that the separability of classes could be improved by a significant amount if information about the class identity was taken into account during learning. This idea was first implemented by the *supervised* SOM [86]. The input vectors were thereby composed of two parts $\mathbf{x}_s$ and $\mathbf{x}_u$, where $\mathbf{x}_s$ is the original input pattern (a spectral or cepstral feature vector), and $\mathbf{x}_u$ is a unit vector with each of its components assigned to one of the phonemic classes [102, p. 161]. The augmented vectors $\mathbf{x} = [\mathbf{x}_s^T, \epsilon\mathbf{x}_u^T]^T$, with ϵ a small factor typically equal to 0.1, were then used as inputs to the SOM. Because the $\mathbf{x}_u$ is the same for vectors of the same class, but different for different classes, clustering of the vectors $\mathbf{x}$ along with the classes is enhanced, which leads to improved class separation. The weight vectors are similarly composed of two parts, $\mathbf{m}_s$ and $\mathbf{m}_u$, respectively: $\mathbf{m} = [\mathbf{m}_s^T, \mathbf{m}_u^T]^T$. During recognition of an unknown $\mathbf{x}$, only the $\mathbf{x}_s$ part is compared with the corresponding $\mathbf{m}_s$ part of the weight vectors in the SOM.

In our newest speech recognition systems we are using the LVQ algorithm [93] when classifying the quasiphonemes. Contrary to the LVQ algorithm the supervised SOM also makes it possible to visualize and monitor the speech signal during use, because the responding areas in the SOM are ordered [90]. The supervised SOM algorithm yields almost the same quasiphoneme classification accuracy as the LVQ algorithm does, and therefore we describe it only in this paper [168], [115].

The SOM used in speech recognition has to be large enough so that all the typical phonemic samples have representations in it. On the other hand, increasing the number of units increases the number of computations needed for classification and also the amount of training data. For a one-speaker system in Finnish we originally had 200–300 units in the SOM classifier. In order to improve classification accuracy, both the number of units and the dimensionality of pattern vectors must be increased; there also must be separate SOM's or LVQ's for stationary and transient phonemes. For real-time operation, special hardware is then needed [115].

The "unit-vector part" $\mathbf{m}_u$ of each weight vector makes it possible to estimate the reliability of the weight vectors formed at the various SOM units. If, during training, a unit consistently produced correct responses to a particular phoneme, the $\mathbf{m}_u$ part of it became rather similar to a unit vector multipled by a constant. On the other hand, if a unit was located in the border zones between the phonemic classes, the $\mathbf{m}_u$ part had several nonzero components. Classification accuracy could then be slightly improved if such unreliable units, with their maximum element being below a given value, were excluded from the SOM. This philosophy does not apply if LVQ is used.

The speech signal is classified every 10 ms into classes corresponding to the stationarity regions of phonemes; the labels thereby produced are called the *quasiphonemes*. The third problem is segmentation of the quasiphoneme strings, and thus of the speech waveform, into standard phonemic transcriptions. Consider that the quasiphonemes are taken

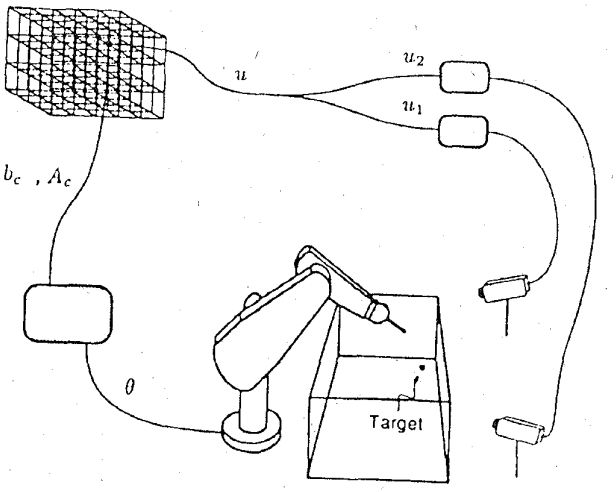


Fig. 11. Visuomotor coordination of robot arm: The image-plane coordinates $\mathbf{u}_1$ and $\mathbf{u}_2$ of the target, as seen by two cameras, are combined into a 4-D camera vector $\mathbf{u}$. The latter serves as input to a 3-D SOM, from which the parameters that define the initial configuration vector $\boldsymbol{\theta}_i$ of the arm are searched. When the true arm coordinates are settled to the value $\boldsymbol{\theta}_i$, a new camera vector $\mathbf{u}_i$ is obtained, which again defines the final configuration vector $\boldsymbol{\theta}_f$.

at regular intervals every 10 ms, whereas the duration of the true phonemes is variable, say, from 40 to 400 ms.

Several different methods can be used for segmentation. In one of the current speech recognition systems the quasiphoneme segmentation is accomplished by the hidden Markov model (HMM) [111]. The basic idea is to consider the quasiphonemes as outputs of states of a stochastic model that undergo transitions according to certain conditional probabilities. The problem is to estimate these probabilities from the available examples. Using the HMM it is then possible to deduce what the most probable phonemes are. We have also variants of our baseline system in which the whole phoneme recognition process is based on Markov models [106].

Dynamical dependencies between the phonemes, due to coarticulation effects in speech, produce systematic errors in the phoneme strings. Some neural networks, like the time-delay neural network (TDNN) [179], take them into account in the phonemic recognition already. In our system the coarticulation errors are corrected at the symbol level of the final transcriptions, which is computationally more effective. The method named the dynamically expanding context (DEC) [88] is an adaptive grammar that automatically learns a great number of correction rules from known examples of pairs of incorrect and correct strings.

D. Other SOM Applications to Pattern Recognition

A general survey of SOM applications to computer vision and texture analysis, including general problems like texture analysis, document processing, medical imaging, remote sensing, target recognition, and speech recognition is given in [102]. The applications include remote sensing, part-family classification, traffic sign recognition, radar classification of sea ice, and face recognition. Some specific reported applications of the SOM have been characterization of paper properties and classification of clouds [157], [178]. SOM-based applications in curve detection have also been reported [186], [134].

V. Robotics

The most sophisticated applications of adaptive control theory are associated with robotics. The artificial neural networks thereby make it possible to describe highly nonlinear and even implicit input-output relations. The SOM has a particular property that can be utilized for the implementation of *arbitrary* input-output relations: since in the ideal case any input state activates exactly one map unit, it is possible to use the SOM in the table-lookup mode, by associating output information with each map unit that can then be read out upon its selection. If the SOM is used for the implementation of control functions, the associated control variables may be set heuristically or control-theoretically in relation to the input states, or they can also be learned adaptively.

A. Visuomotor Control of Robot Arm

In this section we present the adaptive scheme of Ritter *et al.* [145] for the implementation of visuomotor control of a robot arm.

Consider Fig. 11 that illustrates how the coordinates $\mathbf{u}_1$ and $\mathbf{u}_2$ of a target point in the image planes of two cameras are first combined into a four-dimensional (4-D) camera vector $\mathbf{u}$, used as input to a SOM array. This array is 3-D, and it quantifies the working space in an optimal way. The SOM is then used as a table from which the *configuration vector* $\boldsymbol{\theta}$ of the robot arm, i.e., the vector of the future angular coordinates of the arm and its joints is searched.

Quantization of the working space by a reasonably small SOM, however, can only be made in a very coarse way, and $\boldsymbol{\theta}$ must then be interpolated. In addition to the quantized value $\mathbf{b}_c$ of the configuration vector stored in location c, we can also find the "sensitivity matrix" $\mathbf{A}_c$ from that location. One can regard $\mathbf{A}_c$ as the Jacobi matrix that occurs in the first-order term of the Taylor expansion $\boldsymbol{\theta}_i$ of vector $\boldsymbol{\theta}$. If $\mathbf{m}_c$ is the closest codebook vector, the first (initial) linear approximation of the configuration vector is

$$\boldsymbol{\theta}_i = \mathbf{A}_c(\mathbf{u} - \mathbf{m}_c) + \mathbf{b}_c. \tag{17}$$

Even this approximation may be too inaccurate for robot-arm operation, but a corrective step can be made easily. The true configuration vector, obtained by using $\boldsymbol{\theta}_i$ as the control variable, also causes the camera vector to change from $\mathbf{u}$ to $\mathbf{u}_i$. The final configuration vector $\boldsymbol{\theta}_f$ obtained correctively is

$$\boldsymbol{\theta}_f = \boldsymbol{\theta}_i + \mathbf{A}_c(\mathbf{u} - \mathbf{u}_i). \tag{18}$$

Derivation of the adaptation equations of Ritter *et al.* [145] for $\mathbf{A}_c$ and $\mathbf{b}_c$ contains several approximations. Without detailed discussion we recapitulate below the complete

algorithm, in which both the SOM and the control variables are updated simultaneously.

1) Present a randomly chosen target point in the work space.
2) Let the cameras produce the corresponding input signal $\mathbf{u}$ to the SOM.
3) Determine the map unit c corresponding to $\mathbf{u}$.
4) Move the arm to an intermediate position by setting the joint angles to

$$\boldsymbol{\theta}_i = \mathbf{A}_c(\mathbf{u} - \mathbf{m}_c) + \mathbf{b}_c$$

where $\mathbf{A}_c$ and $\mathbf{b}_c$ are found at location c.
5) After the arm has settled to a configuration corresponding to $\boldsymbol{\theta}_i$, execute a correction of the arm position according to

$$\boldsymbol{\theta}_f = \boldsymbol{\theta}_i + \mathbf{A}_c(\mathbf{u} - \mathbf{u}_i)$$

and observe the corresponding camera coordinates $\mathbf{u}_f$.
6) Execute a learning step of the SOM vectors according to

$$\mathbf{m}_r^{\text{new}} = \mathbf{m}_r^{\text{old}} + \epsilon h_{cr}\left(\mathbf{u} - \mathbf{m}_r^{\text{old}}\right)$$

where ϵ is a suitable parameter value.
7) Determine improved values $\mathbf{A}^*$ and $\mathbf{b}^*$ using

$$\mathbf{A}^* = \mathbf{A}_r^{\text{old}} + \delta_2 \cdot \mathbf{A}_r^{\text{old}}(\mathbf{u} - \mathbf{u}_f)(\mathbf{u}_f - \mathbf{u}_i)^T$$
$$\mathbf{b}^* = \mathbf{b}_r^{\text{old}} + \delta_1 \cdot \mathbf{A}_r^{\text{old}}(\mathbf{u} - \mathbf{u}_i)$$

where δ_1 and δ_2 are suitable parameter values.
8) Execute a learning step of the output values stored at map unit c as well its neighbors r

$$\mathbf{A}_r^{\text{new}} = \mathbf{A}_r^{\text{old}} + \epsilon' h'_{cr}\left(\mathbf{A}^* - \mathbf{A}_r^{\text{old}}\right)$$
$$\mathbf{b}_r^{\text{new}} = \mathbf{b}_r^{\text{old}} + \epsilon' h'_{cr}\left(\mathbf{b}^* - \mathbf{b}_r^{\text{old}}\right)$$

where ϵ' is a suitable parameter value, and continue with step 1.

Ritter *et al.* [146], [147], [180] have recently introduced several developments to their basic system, such as the parametrized SOM (PSOM); the main incentives for these solutions have been reduction in parameters and training data, faster learning, completion of partial input vectors by associations, and factoring high-dimensional mappings into a hierarchy of lower-dimensional maps. The "local PSOM" and the "Chebyshev PSOM" [180] are two further developments, which yield still higher accuracies at more effective computation. In particular, these schemes were developed for the learning of the complete grip kinematics of a triple set of three-jointed robot fingers.

B. Robot Navigation

One of the central problems in designing mobile robots and autonomously moving vehicles is the problem of navigation, e.g., avoiding obstacles or staying on the track. Typical motion models and behaviors have to be generated.

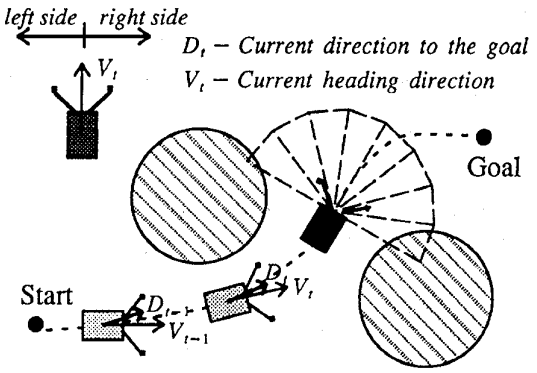


Fig. 12. The simulated robot starts from a point in a landscape with obstacles and tries to reach a given goal point. The robot knows at all times t its own heading direction V_t and the direction to the goal D_t. Using the eight distance sensors, the robot can measure distances to the closest obstacle in the eight directions. Using these and the direction to the goal as inputs to a SOM, the robot reads out the required change in the heading direction ΔV_t. Collisions are detected by the left and right tactile sensors.

In very restricted applications, when the task and the environment are fully specified, it may be possible to plan all the actions beforehand. For more complex domains, however, especially real-world environments where moving obstacles can occur and all the situations to be encountered by the robot cannot be foreseen, it is tedious and even impossible to program the robot in this way.

We review here an approach in which the mapping from situations to actions is learned by the robot using the SOM [60], [61]. Several low-level behaviors like turning at road intersections and avoiding a number of freely placed obstacles can be learned from examples by the SOM. A simulated robot and a simulation environment are used for demonstrations. We assume that the robot has a number of distance sensors pointing in different directions relative to the robot; at any instant of time, each sensor returns the distance to the closest obstacle in its direction (see Fig. 12). In addition, there are two tactile sensors that can detect direct collisions. For desired actions, the robot knows its present heading direction and the absolute direction to a goal, and can also measure changes of its direction given as horizontal angles. Starting from a point on a field having freely placed obstacles, the robot should proceed to the goal without hitting any of the obstacles.

In the first series of experiments described here, the robot was trained by explicitly showing desired trajectories from a number of starting points to a number of goals. Moving along the trajectory shown by the teacher, the robot can measure at each discrete time instant the distance values $(d_1, \cdots, d_8)$ given by its eight distance sensors, the current direction to the goal D_t, and the change of its own orientation (turning angle) from the previous instant, ΔV_t. A collection of these ten-element vectors were used to train the SOM. The SOM network learned the mapping from the nine sensory inputs $d_1, \cdots, d_8, D_t$ to the actions, in this case given by the turning angle ΔV_t. In the training algorithm, it has turned out to be a good strategy to give more weight to training when the robot must turn strongly to

avoid a collision. This can be done by defining the learning-rate parameter α in the SOM learning rule in (7) as an increasing function of the absolute value of ΔV_t.

After training, the SOM can be used to control robot motion. At a particular time, the distance measurements and the direction to the goal are fed to the map, the best-matching unit is found based on these weight vector elements, and information stored at the best-matching unit is used to specify the change of orientation. Based on this, the robot computes the new heading direction and proceeds. The robot is assumed to move at a constant velocity.

The performance of the robot trained with the above procedure depends crucially on how representative the training set is: if the robot encounters a situation for which no training samples were presented, the behavior is unpredictable. To alleviate this problem, a trial-and-error or reinforcement-type learning scheme was employed, which allows continuous learning while the robot is operating [60]. If the robot collides with an obstacle, which is detected by the two tactile sensors, the robot will backtrack and learn an improved trajectory that will avoid a collision in a similar situation in the future. In practice the robot always stores a number M of the latest values of the turning angle ΔV_t, $t = 0, -1, \cdots, -M + 1$ in a buffer, allowing it to backtrack M steps. After this, an incremental value is added to the last element of the weight vectors, corresponding to the input ΔV_t, as if the robot had actually turned to avoid the obstacle. The incremental value is deterministic and suitably adjusted to guarantee smooth behavior after learning.

In a simulation [60], 20 paths with three obstacles were first explicitly shown to the robot and a 128-unit SOM was trained in the way described above. Afterwards the robot was allowed to run 500 different navigation trials from starting positions to goals in a landscape with six obstacles, with the reinforcement learning taking place all the time. Only a total of two collisions occurred during the 500 runs.

C. Other SOM Applications to Robotics

There exist several achievements in robotics ensuing from the SOM research. Problems in machine vision were already surveyed in Section IV-D. In addition to the control of the robot arm with its visuomotor coordination, and collision-free navigation surveyed above, some further problems in robotics are intersensory coordination in goal-directed movements, learning of motor shapes, and optimal information storage for robot control [102].

VI. Telecommunications

Modern telecommunications, especially by electromagnetic waves, can benefit of neural-network solutions in many ways. In this section, we present a few examples in which the SOM principle has been used to improve the separability of quantized signal states, to equalize channel properties, to reduce co-channel interference, and to compress information.

A. Adaptive Detection of Quantized Signals

We start this section with a simple [91] principle that may better apply to wired than wireless telecommunications, for instance, when using a power line or carrier cable for transmission. The transmission of signals from the transmitter to the receiver thus occurs through a single path, not via multiple (reflected) paths as usual in radio traffic. Information, for instance of analog signal values, is converted into a sequence of discrete codes or symbols, each of which is transmitted as a quantized analog signal level or discretized amplitude, frequency, or phase modulation of a carrier wave. No intersymbol or co-channel interference is yet considered in this section; we shall revert to these problems in Sections VI-B–C. In other words, the main type of disturbance consists of nonlinear and time-variant changes in modulation, caused by the transmitting and receiving electronics and variable attenuation, and white noise.

For brevity of discussion we only consider the quadrature amplitude modulation (QAM) here, and Sections VI-B–C then continue the discussion. In the QAM, the transmission channel is divided into two independent subchannels (defined as the "in-phase" (I) and "quadrature" (Q) components of the transmitted wave, respectively): two carrier waves that have the same frequency but a relative phase of 90° are transmitted through the same frequency channel and amplitude-modulated independently. In the 16QAM there are four discrete-valued modulation levels in each subchannel. The ideal signal constellation in the 2-D I-Q-coordinate lattice is depicted in Fig. 13(a). Thus 16 symbols can be encoded and decoded.

The square lattice formed of the quantized signals may be deformed during use in many ways, as depicted in Fig. 13: saturable nonlinearity of the frequency channel (b), and changing of the relative phase of the subchannels (c). Noise can be superimposed on each signal level. All the deformations may be time variable. We shall show that a SOM, located at the receiving end, can be used as an effective adaptive decoder for the QAM.

Consider that the I and Q signals at the receiving end, representing the amplitude modulation of the zero and 90° phase components, respectively, are regarded as the two input signals to the SOM. The 2-D input vector $\mathbf{x}$ is then clustered, each cluster corresponding to one symbol in an orderly fashion. As the input signal selects the closest unit in a SOM, detection of the signal state is made in a noise-tolerant way. The $\mathbf{m}_i$ can be updated for maximum separation of the signal states.

Each of the nodes or units of the 4 by 4 SOM is expected to be able to follow up its own cluster under deformations of the types depicted in Fig. 13(b) and (c), and their mixtures.

As the weight vectors of the SOM algorithm are supposed to follow the clusters all the time, the learning-rate factor α in (7) should be kept constant and at a fairly small value.

There exists one problem, not yet discussed, associated with every SOM. Although the $\mathbf{m}_i$ tend to follow the probability density function $p(\mathbf{x})$, boundary effects at the

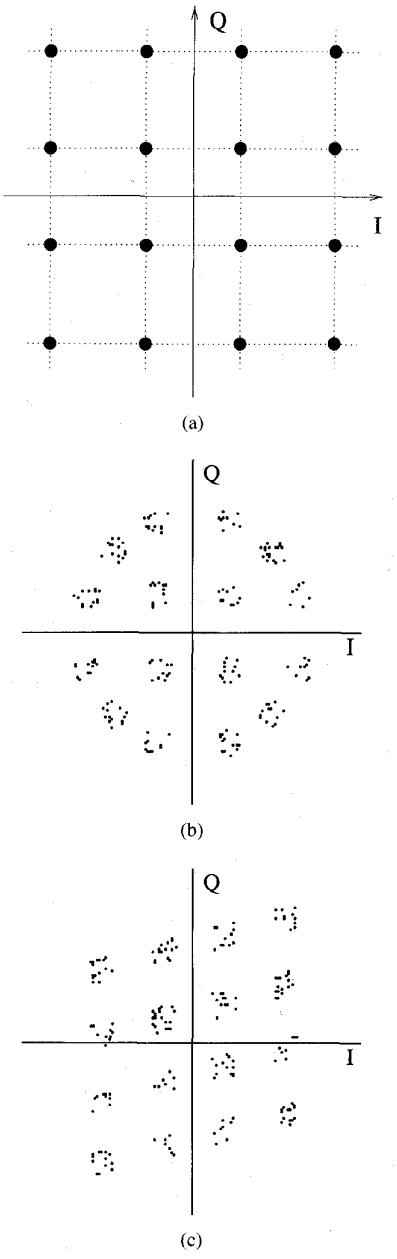


Fig. 13. Ideal signal constellation of the 16QAM used in digital communication systems where the in-phase (I) and quadrature (Q) components occupy discrete lattice points in the (a) I-Q-coordinate system. Typical nonlinear distortions occuring in practical systems using QAM modulation are the (b) corner and (c) lattice collapse situations.

edge units of the SOM usually cause the $\mathbf{m}_i$ lattice to be *contracted*, and this effect, in a small lattice and with low input dimensionality (two) of the vectors can be severe. Without lengthy discussion [102], [101] it may be mentioned that the simplest way to compensate for such a contraction, and to make the $\mathbf{m}_i$ follow up $p(\mathbf{x})$ without bias (at least in this example) is to multiply the learning-rate factor α in (7) by a weight W determined according to the following rule. There is a different weight $W = 1$ for the inside units of the lattice, $W = W_1$ for the corner

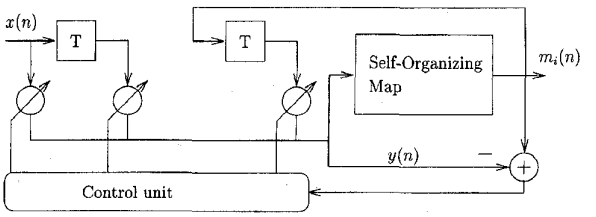


Fig. 14. Block diagram of the neural equalizer where the traditional DFE is used in cascade with the SOM. The DFE is first used to correct linear distortions and intersymbol interference due to the channel dynamics, while the SOM adapts to nonlinear distortions, changing the decision levels of the signal constellation.

units of the SOM, and $W = W_2$ for the edge units that are not corner units, respectively.

1) Weighting Rule: The value W_1 is applied when updating the selected weight vector (but not any other of its topological neighbors) and if the value of $\mathbf{x}$ is in one of the four "outer corner sectors," i.e., outside the array and such that one of the corner units is closest.

The value W_2 is applied if both of the following conditions are simultaneously satisfied: 1) The value of $\mathbf{x}$ lies outside the array, but the closest $\mathbf{m}_i$ does not belong to any corner unit, and 2) This selected edge unit or any of its topological neighbors, which must be one of the edge units (eventually even a corner unit) is being updated.

In order that the $\mathbf{m}_i$ approximate the clusters in an approximately unbiased way, we may choose $W_1 = 81$, $W_2 = 9$, when the neighborhood function is defined by the neighborhood set consisting of the closest neighbors of the "winner," and the density function is rather uniform. Notice that we must have $W\alpha \leq 1$ at all times; unless α cannot be selected small enough, the *Batch Map* mode of the SOM, which does not contain any learning-rate factor α, must be used.

We shall omit here all experiments on this simple system; however, related behavior is discussed [102], [101]. We revert to the performance analysis with more complete systems in Sections VI-B–C.

B. Channel Equalization in the Adaptive QAM

The so-called *linear transversal equalizers* or *decision-feedback equalizers* (DFE) [138] are able to adapt to linear distortions and intersymbol interference of discrete signal levels. However, they are generally not performing well under nonlinear distortions. To combine the advantages of the conventional methods and the SOM-based adaptive detection described in the previous section, we have developed nonlinear dynamic adaptation methods [94]–[96], [100], [68]. In these methods, called the *neural equalizers*, the SOM is used either in cascade or in parallel with the conventional equalizer.

The block diagram of the cascade combination of the DFE structure with the SOM is shown in Fig. 14. The $\mathbf{m}_i$ values of the SOM define the adaptive decision levels in the detector part. Let $\mathbf{y}(n)$ be the output from the channel equalizer (DFE) part for sample n. The output error, $\boldsymbol{\epsilon}(n) = \mathbf{y}(n) - \mathbf{m}_i(n)$, controls the adaptation of

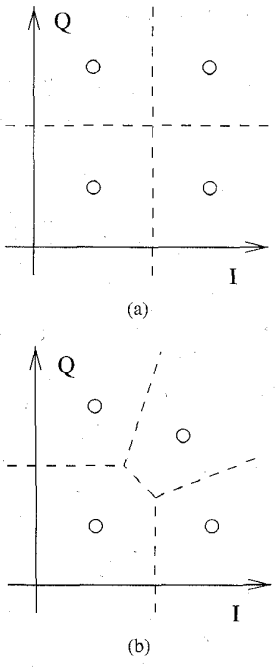


Fig. 15. Decision levels (a) in the ideal situation and (b) after adaptation to the corner collapse distortion.

the DFE. The basic principle of this neural equalizer is that the DFE corrects the linear distortions and the channel dynamics, while the SOM is used to adaptively compensate for nonlinearities.

A typical nonlinear error is the "corner collapse" situation illustrated in Fig. 15. The decision levels between the lattice points of the SOM are then adaptively moved according to the received signal samples, as shown by the dashed lines for one quarter of the signal constellation.

As the theory of the conventional DFE is well documented, we shall omit here the explanation of how the linear-digital-filter coefficients are updated.

We have tested the neural equalizer structures with extensive simulations using the distortions shown in Fig. 13(b) and (c). The communication channel was simulated using a two-path model [67]. In this model one reflected wave, with the reflection coefficient equal to -0.2, was superimposed on the primary wave. The signal-to-noise ratio (SNR) was varied in the simulations. Also the start-up behavior of the neural equalizer was tested.

An example of these simulations is shown in Fig. 16. The effect of the amount of nonlinear distortion due to corner collapse [Fig. 13(b)], and due to lattice collapse [Fig. 13(c)] situation have been investigated. In the corner collapse situation the corners collapse from 0 to 1.5 units, the diagonal distance between neighboring lattice points being defined as two units. In the lattice collapse simulation the horizontal axis of the signal constellation lattice rotates from 0 to 40°. The simulation results in Fig. 16 clearly show that with different SNR the neural equalizer is able to tolerate nonlinearities much better than the conventional equalizer. In addition to the above examples, the synchronization and start-up behaviors of the neural equalizers have been studied [141]. We have also studied more complex channel models.

The effects of distortions caused by noise and nonlinear fading have been studied using the Batch Map version of the SOM algorithm in [98], [101], and [142].

In general, the simulations have shown that the SOM-based neural equalizer outperforms the conventional equalizer structures in compensating for nonlinear distortions. The performance of the Batch Map based architecture has been found to be the best of the neural equalizers investigated. It should also be mentioned that the computational complexity, i.e., the number of arithmetic operations, of the neural equalizers is comparable to that of the conventional equalizers.

C. Cancellation of Co-Channel Interference

Our reseach for the development of neural receiver architectures has recently been directed toward the cancellation of interference effects between channels [143]. The active learning capability of the SOM can be utilized to follow up the distributions of the interfering signals using various repetitive structures of the interference. The main advantage of this approach is that the receiver can adapt to the interference without any prior knowledge of the characteristics of the interfering signal.

The block diagram of the receiver architecture with the SOM-based interference cancellation is shown in Fig. 17. In this structure the DFE is used to compensate for the intersymbol interference and multipath propagation. The co-channel interference is cancelled using the feedback loop that contains a SOM. The error signal e_1 corresponding to the distortion, i.e., the interference and noise, is used in estimating the next value of the error for compensation. The SOM is able to adapt to the characteristics of the co-channel interference from sample to sample. Properties of this interference sequence are used in calculating the new interference estimate.

Two variations of the interference estimation algorithm have been developed: 1) Each error sample is classified separately and 2) error samples are combined into a sequence vector that is used in classification. The structures of the compensation algorithms, called the sep-SOM and the con-SOM, respectively, are depicted in Fig. 17(a) and (b). Detailed descriptions of both algorithms can be found in [143]. Also the computational complexities, which strongly depend on the size of the SOM used for error classification, are discussed in [143].

We have studied the performance of the sep-SOM and con-SOM structures by simulations using the 16QAM modulation and a simplified one-path channel model. Different types of interferences were investigated: 1) interference caused by a Gaussian noise source and 2) interfering signal from another transmission channel. The simulation results obtained so far have been very promising. We were able to show that the performance of the receiver can be generally improved using the SOM-based co-channel interference cancellation.

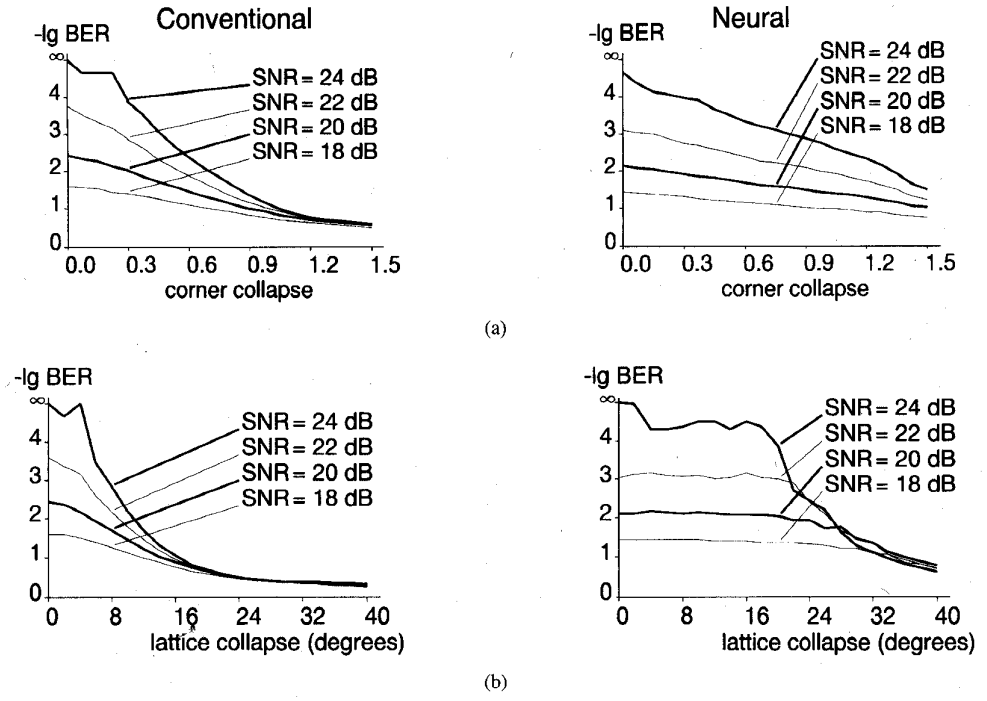


Fig. 16. Performance comparisons between the conventional equalizer (images on the left) and the neural equalizer (images on the right) for two types of nonlinearities are illustrated here. The ordinate value is $-\lg$ BER (negative logarithm of bit error ratio): the higher the curve, the better the error tolerance. (a) The corner collapse situation where the corners collapse from 0 to 1.5 units, the diagonal distance between neighboring points being defined as two units. (b) The horizontal axis of the signal constellation lattice rotates from 0 to 40°. The simulations clearly show that in both cases the neural equalizer is able to compensate for the nonlinearity much better than the conventional equalizer. The SNR (in decibels) has been varied in all simulations.

D. Encoding and Decoding of Images by a Pair of SOM's

In the previous example, the signal states defined by the QAM were assumed to form a regular array, and the codes were assumed to be pseudorandom vectors, their probability density function at the receiving end consisting of 16 clusters of equal probability mass.

In this section we discuss another example in which the input information is not representable by separable clusters, and its optimal coding ought to be derivable from the density function. We solve this problem by having one SOM at the transmitting end as the encoder, and an identical SOM at the receiving end as the decoder, respectively.

As the practical example we consider *image compression* for its error-tolerant transmission. Like in the previous example, information is transmitted in the form of discrete symbols. These symbols could be modulated, for instance, by the QAM, and if there were deformations of the (quantized) signal states during their transmission, they could be compensated for by the SOM as discussed in Sections VI-A–C. The QAM, however, is only a pair of stages for the compensation of channel properties, whereas a further SOM is needed for the replication of the image. In the following discussion we omit the adaptive compensation and equalization of the channel and concentrate on the replication problem. Transmission of the codes is also different from what was indicated above.

Encoding or image compression follows the idea of VQ [123], which is a powerful compression method, but usually rather sensitive to codeword errors. It has been shown elsewhere [124], [117], [30] that the VQ codebooks produced by the SOM algorithm are comparable in *quantization accuracy* to codebooks designed by the well-known LBG method [53].

We used graylevel images in all examples discussed. For VQ the images were divided into nonoverlapping blocks of 4 by 4 pixels, corresponding to 16-D sample vectors. In the following experiment, a set of 28 images was used for training, and an image, which was not included in the training set, was used to test the system accuracy and efficiency.

The main problem with traditional VQ is that if the codeword y is transmitted through a noisy transmission channel, the errors may change y to some other codeword y'. If $\mathbf{x}$ is the sample vector corresponding to y and $\mathbf{x}'$ that corresponding to y', respectively, in a *nonordered* codebook $\mathbf{x}$ and $\mathbf{x}'$ may look completely different: the erroneous codeword will thus be decoded into a whole block of completely erroneous pixels.

The SOM trained by representative sample vectors has the wanted property that the values of the weight vectors $\mathbf{m}_i$ after training are *spatially ordered* in the array, i.e., the neighboring $\mathbf{m}_i$ vectors in any place and any direction of the SOM array are more similar than the more remote

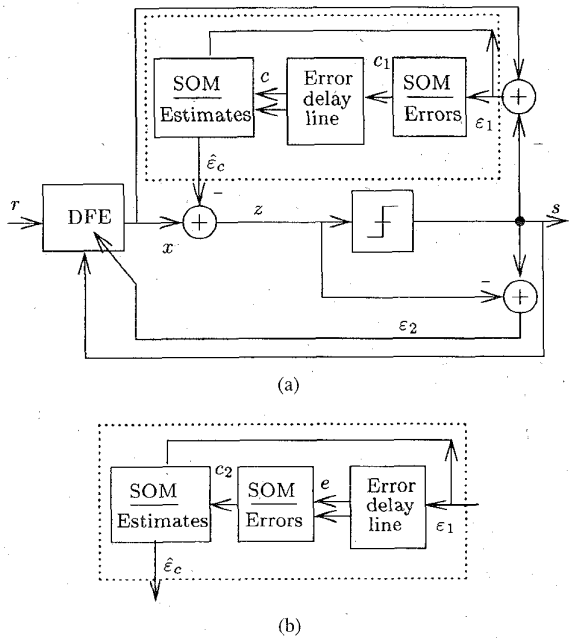


Fig. 17. Architecture of the new SOM-based receiver. The DFE is used for cancellation of intersymbol interference and multipath propagation. Other types of interferences, e.g., co-channel interference, are cancelled by estimating the error signal in the feedback loop. The SOM is able to adapt to the characteristics of the distortion by finding the interference from sample to sample. Two variations of the compensation algorithm for interference estimation are depicted: (a) Each error sample is processed and classified separately (sep-SOM) and (b) error samples are concatenated into a sequence vector that is then classified (con-SOM).

ones. This gives the idea of using the *array coordinates of the SOM as the codes*. As the $\mathbf{m}_i$ also tend to approximate the density function of the sample vectors $\mathbf{x}$, distribution of the codes so obtained into the input domain is made in an almost optimal way.

Encoding thus means conversion of the new input vector, corresponding to the block of pixels to be transmitted, into the discrete *array coordinates of the winner* in the SOM, that are then transmitted in the form of one or several codes. At the receiving end these codes select the coordinates of an *identical SOM array*, and the weight vector $\mathbf{m}_i'$ at that location is used as the replica of the $\mathbf{m}_i$, or the original block. This is the decoding process.

If now the transmitted code y is changed into another code y' due to noise, ordering of the $\mathbf{m}_i'$ guarantees that if y' is similar to y, the decoded sample vector $\mathbf{x}'$ will also be similar to the transmitted sample vector $\mathbf{x}$.

In [112] error-theoretic considerations were applied to the design of a codebook for a noisy environment. The algorithm (where the training neighborhood was related to the likelihood of changing the selected codeword to other codewords) turned out to be almost identical with the SOM training algorithm. *The SOM algorithm therefore leads to a codebook in which the distortion effects are automatically taken into account.*

The benefits of an error-tolerant codebook in data transmission have been demonstrated in [14], in an experiment where speech amplitude values were quantized and random bit substitutions for codewords could happen. The SOM algorithm gave a 3 dB reduction in the replication distortion compared to random codeword schemes. We will demonstrate much higher improvements in image transmission.

To demonstrate the performance of a SOM-based error-tolerant image compression system against transmission channel errors, two transmission coding schemes were used [75], [76]. (The codebook design method itself is general and does depend on these schemes.) In the first scheme we used the digital pulse amplitude modulation (PAM) model [139, p. 164] with eight possible modulation amplitudes $\{A_m, m = 1, 2, \cdots, 8\}$ (see Fig. 18). The errors in the PAM model are amplitude level changes due to channel noise. With the noise contents discussed in this work, especially if the noise is normally distributed, the probability for the noise causing errors over two or more amplitude levels is negligible. Therefore, we only consider one-level errors (up and down) here. The noise is characterized by the probability for the one-level error being equal to p. As the second coding scheme we used a binary symmetric channel (BSC) [139, p. 127]. Again, the probability for an error is denoted by p.

The SOM dimensions for codebooks had to be selected according to the accuracy of representation and principle of transmission. For a block of 16 pixels we decided to use 512 value combinations in all. For the eight-level PAM model, a 3-D SOM had then to be used, because if each dimension of the array has eight units, the total number of units in the SOM is $8^3 = 512$. The coordinate values in each dimension were used to select the amplitude levels in the PAM modulation. The codebook, corresponding to the array coordinates, was then composed of three 3-b addresses of the array. Errors in the transmission channel mean that one or more of the coordinates are wrong by one unit, and a neighboring cell in the SOM array is selected.

For a comparative study with a nonordered codebook, a randomly ordered codebook was prepared by relabeling the codebook vectors. A new, unique random nine-bit codeword was redefined to each.

For the bit-symmetric BSC model the SOM had to be a 9-D hybercube, having $2^9 = 512$ units.

Distortion in the decoded images due to the VQ and transmission errors was measured using the peak SNR (PSNR) defined as

$$\text{PSNR} = 10 \log \frac{255^2}{\text{MSE}} \text{dB} \tag{19}$$

where MSE is the mean-square error. The PSNR values after transmission through the PAM channel with different error probabilities are given in Table 1.

In Fig. 19 two reconstructed images are shown. In these $p = 0.1$. (Because the transmission of three codes was required to transmit one 9-b codeword and the error can happen either upward or downward in amplitude, the probability for an error in codeword $P(p)$ is considerably larger than the probability for an error in a code. Here $P(0.1) = 1.0 - (1.0 - \frac{7}{4} * 0.1)^3 = 0.438$, i.e., more

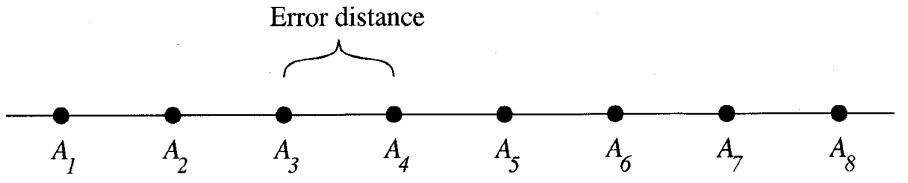


Fig. 18. Signal-space diagram for M-ary PAM signal (here $M = 8$). The distance between successive amplitude levels A_m defines the error probability. If noise amplitude on transmission channel exceeds half of the error distance, some of the codewords may be changed. In this work the noise amplitude probability was taken as normally distributed. At the noise levels discussed in this work, the probability for the noise causing errors corresponding to two or more amplitude levels is negligible.

Table 1 PSNR in Decibels for Different Levels of Noise for the PAM Model. The PSNR Values for Error Free Case are the Same for Both of the Codebooks. When the Probability for Error Increases, the Difference Between the PSNR Values Given by the Ordered and the Nonordered Codebooks, Respectively, Increases Significantly

Probability p for an error	SOM Code	Random Code	Difference in PSNR's
0.0000	31.77	31.77	0.00
0.0001	31.75	31.75	0.24
0.0010	31.48	29.14	2.34
0.0100	30.23	22.42	7.81
0.1000	24.40	13.77	10.63

than 40% of the codewords were erroneous.) In the image with random order in the codewords the errors are usually rather severe; e.g., in the middle of dark areas there are light blocks, and dark blocks are inserted in light areas, respectively. In the image with error-tolerant coding the errors are of different nature: for instance, in the dark area the erroneous blocks are never light, but "almost" dark. The subjectively experienced qualities of the images differ significantly, although the same *number* of codeword errors was present in both of the images.

The PSNR values for BSC channel with different error probabilities are given in Table 2. Like in experiments with the PAM model, the differences in PSNR between ordered and nonordered codebooks are clearly discernible.

E. Other SOM Applications to Telecommunications

It has been shown above that the SOM algorithm may be used to solve certain technical problems in telecommunications. In this section we have collected some other applications to telecommunications.

The SOM algorithm has been used to increase the compression efficiency of VQ by address prediction techniques and SOM codebooks [21], [16], [137], [22], [77], to compensate for transmission error effects (as above) [14], [22], [159], to enhance delta modulation and LPC tandem operations [125], and to utilize the ordering property of the SOM in progressive transmission of vector quantized images [144]. Other applications are in bandwidth compression [55] and adaptive equalization of PAM and QAM signals [136].

The SOM has been used for clone detection [23], [11] and complexity analysis [35] of telecommunications software. Configuration and monitoring of telecommunications traffic networks is a particularly suitable application area for SOM [5], [6], [39]–[43], [165], [166].

VII. Miscellaneous Engineering Applications

In this article we have mainly concentrated on industrial applications of the SOM. In addition to works already referred to in due contexts, we would like to complete the review with the following applications.

A. Measuring and Evaluation Techniques

One major application area of neural networks is combination or *fusion* of signals from different sources such as optical and infrared sensors or sensor arrays. SOM's have thereby been used [1], [47], [48], [184], [110].

Special measurements to which the SOM has been applied are identification of car body steel [80], heating and cooling load estimation [79], calculation of energy losses in distributed systems [182], flow regime identification [18], flow-rate measurement [19], extraction of jet features [50], estimation of torque in switched reluctance motors [46], surface classification [2], [7], [81], odor classification [32], analysis of particle jets [13], detection of particle-impact noise [151], estimation of shear velocity [17], intrusion detection [38], assessment of composite damages [54], evaluation of solid print quality [177], and beer-quality grading [20].

B. Designing and Testing Methods

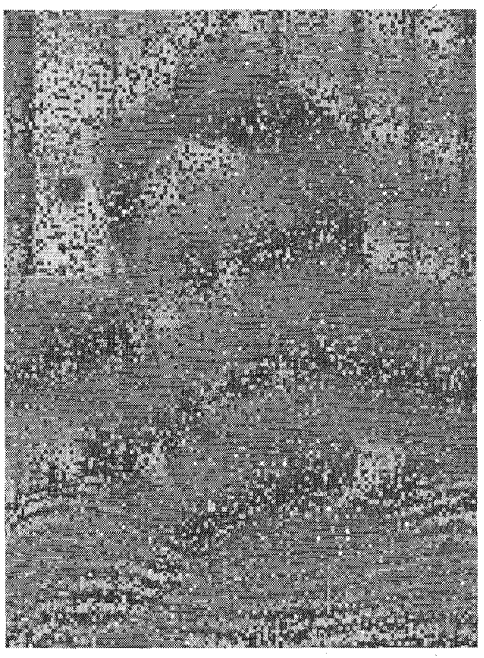
Very large scale integration (VLSI) of electronic circuits is a complex problem that involves tasks ranging from architectural considerations to testing.

On the more abstract level the SOM has been used for the determination of optimal digital coding of information [71] and synthesis of synchronous digital systems [63]–[66].

For the optimization of circuit layout and component placement, modified SOM algorithms have been used [25], [72], [73], [24], [26], [62], [82], [83], [140], [149], [154]–[156], [164], [170], [193]–[197].

Analysis of process data in the VLSI fabrication process is a very important application to which SOM has introduced useful design tools [116].

In testing, SOM methods have been used to eliminate defective cells [190] and to detect partial discharge sources in the circuit [152].



(a)

(b)

Fig. 19. The encoded and decoded images after transmission through a very noisy ($p = 0.1$) channel. Image (a) has been vector quantized with a nonordered codebook and the image (b) with an ordered codebook, respectively. The subjectively experienced qualities of the images differ significantly, although the same *number* of codeword errors was present in both images.

VIII. Conclusion

The number of practical applications of the SOM has been increasing at a rate of about 15% in recent years.

The most straightforward results in engineering have been obtained in the analysis of industrial processes, where, for instance, characteristic states in the operation, otherwise very difficult to discern, have been discovered.

Table 2 PSNR in Decibels for Different Levels of Noise for BSC Channel

Probability p for an error	SOM Code	Random Code	Difference in PSNR's
0.0000	31.63	31.63	0.00
0.0001	31.56	31.12	0.44
0.0010	30.72	28.24	2.48
0.0100	26.20	21.33	4.87
0.1000	17.67	13.01	4.66

An example of hybridization of the SOM and the back-propagation algorithm is a system developed to define paper quality. This system has been adopted to practice some years ago.

Important progress in the use of the SOM has taken place in computer vision. For instance, in the timber industry, board quality assessment systems based on the SOM have been in use in a real plant for half a dozen years. Determination of the formation of paper by analyzing its texture is another application already realized years ago. Robot vision results, on the other hand, have still been demonstrated in experimental environments only.

Application of the SOM to VLSI circuit fabrication, ranging from design to the control of the foundry and also involving the testing phase has been under intense investigation during recent years. Most of the studies have been done in experimental plants that very much resemble the industrial production plants.

Robotics is a promising area for the SOM-based methods; progress in it has mainly been due to one or two strong research groups. The main advantages when using the SOM are the possibility to define very nonlinear input-output relations for multivariable (multijoint) systems, and creation of structured internal representations of the environment as well as of the robot itself in the SOM or a system of SOM's.

Another very promising area for SOM-based methods is in telecommunications at large, and mobile communications in particular; the first steps have just been taken, whereby it is not yet possible to report the performance of complete working systems.

One interesting question is whether the SOM needs special computer hardware. Recent development of microprocessors, signal processors, and workstation computers has been so rapid that theory, software, and hardware have kept well in pace with each other. Only in very demanding tasks such as real-time adaptive computer vision one might need special hardware, but in these tasks also much basic research is still needed to specify the transforms more closely.

We have encountered only few specific applications with several hundred inputs and a couple thousand SOM units, where experimenting with such large SOM's would have been too time consuming, unless we had used a massively parallel SIMD computer. The same experiments, however, might have been carried out equally fast on traditional computers using some more sophisticated shortcuts of the SOM

algorithm. However, the true accuracy of these shortcuts has not yet been quite clear to us. With such a dimensionality of the SOM the amount of necessary training data will already be a problem, whereas good practical results can often be obtained with significantly smaller maps. Thus the need for special hardware for the SOM is not yet quite urgent.

REFERENCES

[1] E. R. Addison and W. Dedmond, "Criteria for choosing connectionist paradigms for real-time data fusion and adaptive discrimination," *Neural Networks*, vol. 1, 1 suppl., p. 419, 1988.

[2] J. T. Alander *et al.*, "Surface type recognition by a hair sensor," in *Communication Control and Signal Processing*, E. Arikan, Ed. Amsterdam: Elsevier, 1990, pp. 1757–1764.

[3] ——, "Process error detection using self-organizing feature maps," in *Artificial Neural Networks*, T. Kohonen *et al.*, Eds. Amsterdam: North-Holland, 1991, vol. 2, pp. 1229–1232.

[4] ——, "Process error detection using self-organizing feature maps," Res. Reps. A5, Rolf Nevanlinna Inst., Helsinki, Finland, 1991.

[5] N. Ansari and Y. Chen, "A neural network model to configure maps for a satellite communication network," in *Proc. GLOBECOM'90, IEEE Global Telecommun. Conf. and Exhibit.: Communications: Connecting the Future*, Piscataway, NJ, 1990, vol. 2, pp. 1042–1046.

[6] N. Ansari and D. Liu, "The performance evaluation of a new neural network based traffic management scheme for a satellite communication network," in *Proc. GLOBECOM'91, IEEE Global Telecommun. Conf. Countdown to the New Millennium: Personal Communications Services (PCS)*, Piscataway, NJ, 1991, vol. 1, pp. 110–114.

[7] A. Autere *et al.*, "Surface type recognition by a hair sensor," Res. Rep. A2, Rolf Nevanlinna Inst., Helsinki, Finland, 1990.

[8] N. R. Ball and K. Warwick, "Application of augmented-output self organizing feature maps to the adaptive control problem," in *Proc. INNC'90, Int. Neur. Network Conf.*. Dordrecht, The Netherlands: Kluwer, 1990, vol. 1, p. 242.

[9] N. Ball *et al.*, "Neural networks for power systems alarm handling," *Neurocomputing*, vol. 4, no. 1–2, pp. 5–8, 1992.

[10] D. Ballard and C. Brown, *Computer Vision*. Englewood Cliffs, NJ: Prentice-Hall, 1982.

[11] P. Barson *et al.*, "Dynamic competitive learning applied to the clone detection problem," in *Proc. Int. Workshop on Applications of Neural Networks to Telecommun.*, 2nd ed. Hillsdale, NJ: L. Erlbaum, 1995, pp. 234–241.

[12] T. Baumann, A. Germond, and D. Tschudi, "Impulse test fault diagnosis on power transformers using Kohonen's self-organizing neural network," in *Proc. 3rd Symp. on Expert Syst. Applicat. to Power Syst.*, Tokyo and Kobe, 1991.

[13] K.-H. Becks, J. Dahm, and F. Seidel, "Analysing particle jets with artificial neural networks," in *Industrial and Engineering Applicat. of Artificial Intell. and Expert Syst. 5th Int. Conf. IEA/AIE-92*, F. Belli and F. J. Radermacher, Eds. Berlin, Heidelberg: Springer, 1992, pp. 109–112.

[14] D. S. Bradburn, "Reducing transmission error effects using a self-organizing network," in *Proc. IJCNN'89, Int. Joint Conf. on Neur. Networks*, Piscataway, NJ, 1989, vol. 2, pp. 531–537.

[15] J. Buhmann, J. Lange, and C. von der Malsburg, "Distortion invariant object recognition by matching hierachically labeled graphs," in *Proc. IJCNN'89, Int. Joint Conf. on Neur. Networks*, Piscataway, NJ, 1989, vol. 1, pp. 155–159.

[16] G. Burel and J.-Y. Catros, "Image compression using topological maps and MLP," in *Proc. ICNN'93, Int. Conf. on Neur. Networks*, Piscataway, NJ, 1993, vol. 2, pp. 727–731.

[17] P. Burrascano, P. Lucci, G. Martinelli, and R. Perfetti, "Shear velocity estimation by the combined use of supervised and unsupervised neural networks," in *Proc. ICASSP'90, Int. Conf. on Acoust., Speech and Signal Process.*, Piscataway, NJ, 1990, vol. 4, pp. 1921–1924.

[18] S. Cai, H. Toral, and J. Qiu, "Flow regime identification by a self-organizing neural network," in *Proc. ICANN'93, Int. Conf. on Artificial Neural Networks*, S. Gielen and B. Kappen, Eds. London: Springer, 1993, p. 868.

[19] S. Cai and H. Toral, "Flowrate measurement in air-water horizontal pipeline by neural network," in *Proc. IJCNN-93-Nagoya, Int. Joint Conf. on Neur. Networks*, Piscataway, NJ, 1993, vol. 2, pp. 2013–2016.

[20] Y. Cai, "The application of the artificial neural network in the grading of beer quality," in *Proc. WCNN'94, World Congress on Neur. Networks*. Hillsdale, NJ: L. Erlbaum, 1994, vol. 1, pp. 516–520.

[21] S. Carrato, G. L. Siguranza, and L. Manzo, "Application of ordered codebooks to image coding," in *Proc. 1993 IEEE Workshop, Neur. Networks for Signal Process.*, Piscataway, NJ, 1993, pp. 291–300.

[22] S. Carrato, "Image vector quantization using ordered codebooks: Properties and applications," *Signal Process.*, vol. 40, no. 1, pp. 87–103, 1994.

[23] S. Carter, R. J. Frank, and D. S. W. Tansley, "Clone detection in telecommunications software systems: A neural net approach," in *Proc. Int. Workshop on Applicat. of Neur. Networks to Telecommun.* Hillsdale, NJ: L. Erlbaum, 1993, pp. 273–287.

[24] D. D. Caviglia *et al.*, "Pre-placement of VLSI blocks through learning neural networks," in *Proc. EDAC, Europe. Design Automat. Conf.*. Glasgow, U.K./Washington, DC: IEEE Comput. Soc., 1990, pp. 650–654.

[25] R.-I. Chang and P.-Y. Hsiao, "Force directed self-organizing map and its application to VLSI cell placement," in *Proc. ICNN'93, Int. Conf. on Neur. Networks*, Piscataway, NJ, 1993, vol. 1, pp. 103–109.

[26] ——, "Force directed self-organizing maps for L-shaped cell placement using delta learning rule," in *Proc. ICNN'94, Int. Conf. on Neur. Networks*, Piscataway, NJ, 1994, pp. 3381–3386.

[27] Y. Chen, "Artificial neural networks and their applications in control and system engineering: An introduction of neural networks," *Power Syst. Technol.*, no. 1, pp. 56–58, Jan. 1993.

[28] R. W. Conners and C. A. Harlow, "Equal probability quantizing and texture analysis of radiographic images," *Computer Graphics and Image Process.*, vol. 8, no. 3, pp. 447–463, Dec. 1978.

[29] ——, "A theoretical comparison of texture algorithms," *IEEE Trans. Patt. Anal. and Mach. Intell.*, vol. PAMI-2, pp. 204–222, May 1980.

[30] J. A. Corral, M. Guerrera, and P. J. Zufiria, "Image compression via optimal vector quantization: A comparison between SOM, LBG and k-means algorithms," in *Proc. ICNN-94, Int. Conf. on Neur. Networks*, IEEE, Piscataway, NJ, 1994, vol. 6, pp. 4113–4118.

[31] P. Cosi, G. De Poli, and G. Lauzzana, "Timbre classification by NN and auditory modeling," in *Proc. ICANN'94, Int. Conf. on Artif. Neur. Networks*, M. Marinaro and P. G. Morasso, Eds. London: Springer, 1994, vol. 2, pp. 925–928.

[32] F. Davide, C. Di Natale, and A. D'Amico, "Sensor arrays and self-organizing maps for odour analysis in artificial olfactory systems," in *Proc. ICANN'94, Int. Conf. on Artif. Neur. Networks*, M. Marinaro and P. G. Morasso, Eds. London: Springer, 1994, vol. 1, pp. 354–357.

[33] S. B. Davis and P. Mermelstein, "Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences," *IEEE Trans. Acoust., Speech Signal Process.*, vol. ASSP-28, no. 4, pp. 357–366, 1980.

[34] L. Ding, J. Li, and Y. Xi, "Generalized self-organized learning in neural network modeling for nonlinear plants," *Acta Electronica Sinica*, vol. 20, no. 10, pp. 56–60, Oct. 1992.

[35] S. Field, N. Davey, and R. Frank, "A complexity analysis of telecommunications software using neural networks," in *Proc. 2nd Int. Workshop on Applicat. of Neural Networks to Telecommun.* Hillsdale, NJ: L. Erlbaum, 1995, pp. 226–233.

[36] F. Firenze, L. Ricciardiello, and S. Pagliano, "Self-organizing networks: A challenging approach to fault diagnosis of industrial processes," in *Proc. ICANN'94, Int. Conf. on Artificial Neur. Networks*, M. Marinaro and P. G. Morasso, Eds. London: Springer, 1994, vol. 2, pp. 1239–1242.

[37] R. Fischl, "Application of neural networks to power system security: Technology and trends," in *Proc. ICNN'94, Int. Conf. on Neur. Networks*, Piscataway, NJ, 1994, pp. 3719–3723.

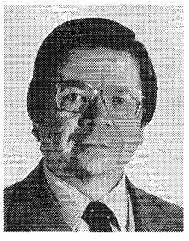
[38] K. L. Fox, R. R. Henning, J. H. Reed, and R. P. Simonian, "A neural network approach toward intrusion detection," in *Proc. 13th Natl. Computer Security Conf. Inform. Syst. Security Standards—The Key to the Future*. Gaithersburg, MD: NIST, 1990, vol. 1, pp. 124–134.

[39] T. Fritsch and W. Mandel, "Communication network routing using neural nets-numerical aspects and alternative approaches," in *Proc. IJCNN'91 Int. Joint Conf. on Neur. Networks*, Piscataway, NJ, 1991, vol. 1, pp. 752–757.
[40] T. Fritsch, M. Mittler, and P. Tran–Gia, "Artificial neural net applications in telecommunication systems," *Neur. Comput. and Applicat.*, vol. 1, no. 2, pp. 124–146, 1993.
[41] T. Fritsch, P. H. Kraus, H. Przuntek, and P. Tran–Gia, "Classification of Parkinson rating-scale-data using a self-organizing neural net," in *Proc. ICNN'93, Int. Conf. on Neur. Networks*, Piscataway, NJ, 1993, vol. 1, pp. 93–98.
[42] T. Fritsch and S. Hanshans, "An integrated approach to cellular mobile communication planning using traffic data prestructured by a self-organizing feature map," in *Proc. ICNN'93, Int. Conf. on Neur. Networks*, Piscataway, NJ, 1993, vol. 2, pp. 822D–822I.
[43] T. Fritsch, "Cellular mobile communication design using self-organizing feature maps," in *Neural Networks in Telecommunications*. Dordrecht, The Netherlands: Kluwer, 1994, pp. 211–232.
[44] H. Furukawa, T. Ueda, and M. Kitamura, "A systematic method for rational definition of plant diagnostic symptoms by self-organizing neural networks," in *Proc. 3rd Int. Conf. on Fuzzy Logic, Neur. Nets. and Soft Computing*, Iizuka, Japan, Fuzzy Logic Syst. Inst., 1994, pp. 555–556.
[45] D. Gabor, "Theory of communication," *J. IEE*, vol. 93, pp. 429–457, 1946.
[46] J. J. Garside, R. H. Brown, T. L. Ruchti, and X. Feng, "Nonlinear estimation of torque in switched reluctance motor using grid locking and preferential training techniques on self-organizing neural networks," in *Proc. IJCNN'92, Int. Joint Conf. on Neur. Networks*, Piscataway, NJ, 1992, vol. 2, pp. 811–816.
[47] K. Gelli, R. A. McLaughlan, R. Challoo, and S. I. Omar, "Multible sensor target classification using an unsupervised hybrid neural network," in *Proc. ICNN'94, Int. Conf. on Neur. Networks*, Piscataway, NJ, 1994, pp. 4028–4032.
[48] ——, "A hybrid neural network architecture for sensor fusion," in *Proc. WCNN'94, World Congress on Neur. Networks*. Hillsdale, NJ: L. Erlbaum, 1994, vol. 1, pp. 679–685.
[49] A. Gersho, "On the structure of vector quantizers," *IEEE Trans. Inform. Theory*, vol. IT-25, pp. 373–380, 1979.
[50] R. Goodacre, "Characterization and quantification of microbial systems using pyrolysis mass spectrometry: Introducing neural networks to analytical pyrolysis," *Microbiol. Europe*, vol. 2, no. 2, pp. 16–22, 1994.
[51] K. Goser, K. M. Marks, U. Rueckert, and V. Tryba, "Selbstorganizierende Parameterkarten zur Prozessüberwachung und—Voraussage," in *3rd Int. GI-Cong. on Wissensbasierte Syst.* München/Berlin/Heidelberg: Springer, Oct. 16–17, 1989, pp. 225–237.
[52] E. Govekar, E. Susič, P. Mužič, and I. Grabec, "Self-organizing neural network application to technical process parameters estimation," in *Artificial Neural Networks*, vol. 2, I. Aleksander and J. Taylor, Eds. Amsterdam: North-Holland, vol. I, 1992, pp. 579–582.
[53] R. M. Gray, "Vector quantization," *IEEE ASSP Mag.*, vol. 1, pp. 4–29, Apr. 1984.
[54] B. Grossman, X. Gao, and M. Thursby, "Composite damage assessment employing an optical neural network processor and an embedded fiber optic sensor array," in *Proc. SPIE—The Int. Soc. for Opt. Eng.*, vol. 1588, pp. 64–75, 1991.
[55] A. Habibi, "Neural networks in bandwidth compression," in *Proc. SPIE—The Int. Soc. for Optical Engineering*. SPIE: Bellingham, WA, 1991, vol. 1567, pp. 334–340.
[56] R. Haralick, K. Shanmugan, and I. Dinstein, "Textural features for image classification," *IEEE Trans. Syst. Man Cybern.*, vol. SMC-3, pp. 610–621, Nov. 1973.
[57] R. M. Haralick, "Statistical and structural approaches to texture," in *Proc. 4IJCPR, Int. Joint Conf. on Patt. Recognit.*, Tokyo: Patt. Recognition Soc. Japan, 1978, pp. 45–69.
[58] ——, "Statistical and structural approaches to texture," *Proc. IEEE*, vol. PROC-67, pp. 786–804, May 1979.
[59] ——, "Statistical image texture analysis," *Handbook of Pattern Recognition and Image Processing*, T. Y. Young and K.-S. Fu, Eds. New York: Academic, 1986.
[60] J. Heikkonen, P. Koikkalainen, and E. Oja, "Self-organizing maps for collision-free nevigation," in *Proc. World Congress on Neur. Networks WCNN*, Portland, OR, 1993, vol. 3, pp. 141–144.
[61] ——, "From situations to actions: Motion behavior learning by self-organization," in *Proc. Int. Conf. on Artif. Neur. Networks ICANN-93*, Amsterdam, 1993, pp. 262–267.
[62] A. Hemani and A. Postula, "Cell placement by self-organization," *Neur. Networks*, vol. 3, no. 4, pp. 337–338, 1990.
[63] ——, "A neural net based self organizing scheduling algorithm," in *Proc. EDAC, Europe. Design Automation Conf.*. Washington, DC: IEEE Comput. Soc. Press, 1990, pp. 136–140.
[64] ——, "Scheduling by self-organization," in *Proc. IJCNN-90-Wash.-DC, Int. Joint Conf. on Neur. Networks*, Piscataway, NJ, 1990, vol. 2, pp. 543–546.
[65] A. Hemani, "High-level synthesis of synchronous digital systems using self-organization algorithms for scheduling and binding," Ph.D. dissertation, Royal Inst. Technol., Stockholm, Sweden, 1992.
[66] ——, "Self-organization and its application to binding," in *Proc. 6th Int. Conf. on VLSI Design*, Bombay, 1993.
[67] J. Henriksson, "Decision directed diversity combiners for digital radio links," Ph.D. dissertation, Acta Polytechnica Scandinavica Electrical Engineering Series No. 54, Helsinki, Finland, 1984.
[68] J. A. Henriksson, K. Raivio, and T. Kohonen, Finnish Pat. 85,548, U.S. Pat. 5,233,635, Australian Pat. 636,494.
[69] Y.-Y. Hsu and C.-C. Yang, "Design of artificial neural networks for short-term load forecasting. I. Self-organizing feature maps for day type identification," in *IEE Proc. C [Generation, Transmission and Distribution]*, vol. 138, no. 5, pp. 407–413, 1991.
[70] H. Hyötyniemi, "Optimal control of dynamic systems using self-organizing maps," in *Proc. ICANN'93, Int. Conf. on Artificial Neural Networks*, S. Gielen and B. Kappen, Eds. London: Springer, 1993, pp. 850–853.
[71] A. C. Izquierdo, J. C. Sueiro, and J. A. H. Mendez, "Self-organizing feature maps and their application to digital coding of information," in *Proc. IWANN'91, Int. Workshop on Artificial Neur. Networks*, A. Prieto, Ed. Berlin, Heidelberg: Springer, 1991, pp. 401–408.
[72] J. W. Jiang and M. Jabri, "A new self-organization strategy for floorplan design," in *Proc. IJCNN'92 Int. Joint Conf. on Neur. Networks*, Piscataway, NJ, 1992, vol. 2, pp. 510–515.
[73] ——, "A new self-organization strategy for floorplan design," in *Proc. ACNN'92, 3rd Australian Conf. on Neur. Networks*, P. Leong and M. Jabri, Eds., Sydney, Australia, Australian Neurosci. Soc.; Australian Telecoms. and Electron. Res. Board; Sydney Univ., 1992, pp. 235–238.
[74] J. Kangas, "Phoneme recognition using time-dependent versions of self-organizing maps," in *Proc. ICASSP-91, Int. Conf. on Acoust., Speech, Signal Process.*, Piscataway, NJ, 1991, pp. 101–104.
[75] J. Kangas and T. Kohonen, "Developments and applications of the self-organizing map and related algorithms," in *Proc. IMACS Int. Symp. on Signal Process., Robotics and Neur. Networks*, Lille, France, 1994, pp. 19–22.
[76] J. Kangas, "Increasing the error tolerance in transmission of vector quantized images by self-organizing map," in *Proc. ICANN'95, Int. Conf. on Artificial Neur. Networks*, Paris, France, EC2&Cie, 1995, vol. 1, pp. 287–291.
[77] ——, "Sample weighting when training self-organizing maps for image compression," in *Neur. Networks for Signal Processing. Proc. 1995 IEEE Workshop*, Piscataway, NJ, 1995, pp. 343–350.
[78] M. Kasslin, J. Kangas, and O. Simula, "Process state monitoring using self-organizing maps," in *Artificial Neural Networks*, vol. 2, I. Aleksander and J. Taylor, Eds. Amsterdam: North-Holland, 1992, pp. 1532–1534.
[79] N. Kashiwagi and T. Tobi, "Heating and cooling load prediction using a neural network system," in *Proc. IJCNN-93-Nagoya, Int. Joint Conf. on Neur. Networks*, Piscataway, NJ, 1993, vol. 1, pp. 939–942.
[80] W. Kessler, D. Ende, R. W. Kessler, and W. Rosenstiel, "Identification of car body steel by an optical on line system and Kohonen's self-organizing map," in *Proc. ICANN'93, Int. Conf. on Artificial Neur. Networks*, S. Gielen and B. Kappen, Eds. London: Springer, 1993, p. 860.

[81] H. Keuchel, E. von Puttkamer, and U. R. Zimmer, "SPIN—Learning and forgetting surface classifications with dynamic neural networks," in *Proc. ICANN'93, Int. Conf. on Artificial Neur. Networks*, S. Gielen and B. Kappen, Eds. London: Springer, 1993, pp. 230–235.
[82] S.-S. Kim and C.-M. Kyung, "Global placement of macro cells using self-organization principle," in *Proc. 1991 IEEE Int. Symp. on Circ. and Syst.* Piscataway, NJ, 1991, vol. 5, pp. 3122–3125.
[83] B. Kiziloglu, V. Tryba, and W. Daehn, "Digital circuit partition by self-organizing maps: A comparison to classical methods," in *Proc. IJCNN-93-Nagoya, Int. Joint Conf. on Neur. Networks*, Piscataway, NJ, 1993, vol. 3, pp. 2413–2416.
[84] T. Kohonen *et al.* "Spectral classification of phonemes by learning subspaces," in *Proc. ICASSP'79*, R. C. Olson, Ed., Piscataway, NJ, 1979, pp. 97–100.
[85] T. Kohonen, *Self-Organization and Associative Memory*, 3rd ed. Berlin/Heidelberg: Springer, 1989.
[86] T. Kohonen, K. Mäkisara, and T. Saramäki, "Phonotopic maps—Insightful representation of phonological features for speech recognition," in *Proc. 7ICPR, Int. Conf. on Patt. Recognit.*, IEEE Computer Soc. Press, 1984, pp. 182–185.
[87] T. Kohonen, "Learning vector quantization for pattern recognition," Rep. TKK-F-A601, Helsinki Univ. Technol., Dept. Techn. Phys., Lab. Computer and Inform. Sci., 1986, p. 18.
[88] ——, "Dynamically expanding context, with application to the correction of symbol strings in the recognition of continuous speech," in *Proc. 8ICPR, Int. Conf. on Patt. Recognition*, IEEE Computer Soc. Press, 1986, pp. 1148–1151.
[89] T. Kohonen, G. Barna, and R. Chrisley, "Statistical pattern recognition with neural networks: Benchmarking studies," in *Proc. IEEE Int. Conf. on Neur. Networks*, San Diego, CA, July 24–27, 1988, vol. 1, pp. 61–68.
[90] T. Kohonen, "The 'neural' phonetic typewriter," *Computer*, vol. 21, no. 3, pp. 11–22, 1988.
[91] ——, Finnish Pat. 83,577, U.K. Pat. 2,233,865.
[92] ——, "Improved versions of learning vector quantization," in *Proc. IEEE Int. Joint Conf. on Neur. Networks*, San Diego, CA, June 17–21, 1990, vol. 1, pp. 545–550.
[93] ——, "The self-organizing map," in *Proc. IEEE*, 1990, vol. 78, pp. 1464–1480.
[94] T. Kohonen *et al.*, "Combining linear equalization and self-organizing adaptation in dynamic discrete-signal detection," in *Proc. IJCNN-90-San Diego, Int. Joint Conf. on Neur. Networks*, 1990, vol. 1, pp. 223–228.
[95] ——, "An adaptive discrete-signal detector based on self-organizing maps," in *Proc. IJCNN-90-WASH-DC, Int. Joint Conf. on Neur. Networks*, 1990, vol. 2, pp. 249–252.
[96] ——, "Performance evaluation of self-organizing map based neural equalizer in dynamic discrete-signal detection," *Artificial Neural Networks, Proc. ICANN-91.* Amsterdam: North-Holland, 1991, vol. 2, pp. 1677–1680.
[97] T. Kohonen, "Self-organizing maps: Optimization approaches," in *Artificial Neural Networks*, T. Kohonen, K. Mäkisara, O. Simula, and J. Kangas, Eds. Amsterdam: North-Holland, vol. 2, pp. 981–990, 1991.
[98] ——, Finnish Pat. 90,705, U.S. Pat. 5,428,644.
[99] T. Kohonen, J. Kangas, and J. Laaksonen, *SOM_PAK, The Self-Organizing Map Program Package*, Helsinki Univ. Technol., Lab. Computer and Information Sci., Espoo, Finland, 1995; "cochlea.hut.fi" (130.233.168.48).
[100] T. Kohonen *et al.* "Start-up behavior of a neural network assisted decision feedback equaliser in a two-path channel," in *Proc. Int. Conf. on Communications*, Chicago, IL, 1992, pp. 1523–1527.
[101] T. Kohonen, "Things you haven't heard about the self-organizing map," in *Proc. ICNN'93, Int. Conf. on Neur. Networks*, 1993, pp. 1147–1156.
[102] ——, "Self-organizing maps," *Springer Series in Information Sciences*, 1995.
[103] ——, "The adaptive-subspace SOM (ASSOM) and its use for the implementation of invariant feature detection," in *Proc. ICANN'95, Int. Conf. on Artificial Neur. Networks.* Paris: EC2&Cie, 1995, vol. 1, pp. 3–10.
[104] T. Kohonen, "Emergence of invariant-feature detectors in self-organization," in *Computational Intelligence: A Dynamic System Perspective*, M. Palaniswami *et al.*, Eds. New York: IEEE Press, 1995, pp. 17–31.
[105] M. Konishi *et al.*, "Application of neural network to operation guidance in blast furnace," in *3rd Europe. Seminar on Neur. Computing: The Marketplace.* London: IBC Tech. Services, 1990, p. 13.
[106] M. Kurimo, "Hybrid training method for tied mixture density hidden Markov models using Learning Vector Quantization and Viterbi estimation," in *Proc. 1994 IEEE Workshop, Neur. Networks for Signal Process. IV*, J. Vlontzos, J.-N. Hwang, and E. Wilson, Eds. New York: IEEE Press, 1994, pp. 362–371.
[107] J. Lampinen and E. Oja, "Self-organizing maps for spatial and temporal AR models," in *Proc. 6th Scandinavian Conf. Image Anal.*, Oulu, Finland, June 19–22, 1989, pp. 120–127.
[108] J. Lampinen and O. Taipale, "Optimization and simulation of quality properties in paper machine with neural networks," in *Proc. ICNN'94, Int. Conf. on Neur. Networks*, Piscataway, NJ, 1994, pp. 3812–3815.
[109] J. Lampinen and E. Oja, "Distortion tolerant pattern recognition based on self-organizing feature extraction," *IEEE Trans. Neur. Networks*, vol. 6, pp. 539–547, 1995.
[110] R. A. Lemos, M. Nakamura, and H. Kuwano, "Applying a self-organizing map to sensor-array characterization," in *Proc. IJCNN-93-Nagoya, Int. Joint Conf. on Neur. Networks*, Piscataway, NJ, IEEE Service Center, 1993, Vol. II, pp. 2009–2012.
[111] S. E. Levinson, L. R. Rabiner, and M. M. Sondhi, "An introduction to the application of the theory of probabilistic functions of a Markov process to automatic speech recognition," *Bell Syst. Tech. J.*, pp. 1035–1073, 1983.
[112] S. P. Luttrell, "Derivation of a class of training algorithms," *IEEE Trans. Neur. Networks*, vol. 1, pp. 229–232, June 1990.
[113] N. Macabrey, T. Baumann, and A. J. Germond, "Load forecasting on an electrical system with the aid of the Kohonen neural network," *Bull. des Schweizerischen Elektrotechnischen Vereins und des Verbandes Schweizerischer Elektrizitätswerke*, vol. 83, no. 5, pp. 13–19, 1992.
[114] J. Makhoul, S. Roucos, and H. Gish, "Vector quantization in speech coding," in *Proc. IEEE*, vol. PROC-73, pp. 1551–1588, 1985.
[115] J. Mäntysalo, K. Torkkola, and T. Kohonen, "Mapping context dependent acoustic information into context independent form by LVQ," *Speech Commun.*, vol. 14, no. 2, pp. 119–130, 1994.
[116] K. M. Marks and K. F. Goser, "Analysis of VLSI process data based on self-organizing feature maps," in *Proc. Neuro-Nîmes, Int. Workshop on Neur. Networks and Their Applications*, Nanterre, France, EC2, 1988, pp. 337–348.
[117] J. D. McAuliffe, L. E. Atlas, and C. Rivera, "A comparison of the LBG algorithm and Kohonen neural network paradigm for image vector quantization," in *IEEE Proc. ICASSP-90, Int. Conf. on Acoust., Speech, Signal Process.*, Piscataway, NJ, 1990, vol. 4, pp. 2293–2296.
[118] K. Möller, "A multiassociative memory for control," in *Proc. ICANN'93, Int. Conf. on Artif. Neur. Networks*, S. Gielen and B. Kappen, Eds. London, Springer, 1993, pp. 593–596.
[119] L. Monostori and A. Bothe, "Convergence behavior of connectionist models in large scale diagnostic problems," in *5th Int. Conf. on Industrial and Engineering Applications of Artif. Intell. and Expert Syst.*, F. Belli and F. J. Radermacher, Eds. Berlin, Heidelberg: Springer, 1992, pp. 113–122, IEA/AIE-92
[120] H. Mori, Y. Tamaru, and S. Tsuzuki, "An artificial neural-net based technique for power system dynamic stability with the Kohonen model," in *Conf. Papers. 1991 Power Industry Computer Application Conf. 17th PICA Conf.*, Piscataway, NJ, pp. 293–301, 1991.
[121] H. Mori, Y. Tamaru, and S. Tsuzuki, "An artificial neural-net based technique for power system dynamic stability with the Kohonen model," *IEEE Trans. Power Syst.*, vol. 7, pp. 856–864, May 1992.
[122] C. Muller *et al.*, "A neural network tool for forecasting French electricity consumption," in *Proc. WCNN'94, World Congress on Neur. Networks.* Hillsdale, NJ: L. Erlbaum, 1994, vol. 1, pp. 360–365.
[123] N. M. Nasrabadi and R. A. King, "Image coding using vector quantization: A review," *IEEE Trans. Comm.*, vol. 36, no. 8, pp. 957–971, 1988.
[124] N. M. Nasrabadi and Y. Feng, "Vector quantization of images based upon the Kohonen self-organization feature maps," *Neur. Networks*, vol. 1, 1 suppl., p. 518, 1988.

[125] J. A. Naylor, "A neural network algorithm for enhancing delta modulation/LPC tandem connections," in *Proc. ICASSP-90, Int. Conf. on Acoust., Speech Signal Process.*, Piscataway, NJ, 1990, vol. 1, pp. 211–224.
[126] D. J. Nelson, S.-J. Chang, and M. Chen, "Modeling the time of occurrence of electric utility peak loads," in *Proc. 1992 Summer Computer Simulation Conf. 24th Annu. Computer Simulation Conf.*, P. Luker, Ed. San Diego, CA: SCS, 1992, pp. 212–217.
[127] D. Niebur and A. J. Germond, "Power system static security assessment using the Kohonen neural network classifier," in *Conf. Papers. 1991 Power Industry Computer Application Conf. 17th PICA Conf.*, Piscataway, NJ, 1991, pp. 270–277.
[128] ——, "Power flow classification for static security assessment," in *Proc. 1st Int. Forum on Applications of Neural Networks to Power Syst.*, M. A. El-Sharkawi and R. J. Marks II, Eds., Piscataway, NJ, 1991, pp. 83–88.
[129] ——, "Unsupervised neural net classification of power system static security states," in *Proc. 3rd Symp. on Expert Syst. Applicat. to Power Syst.*, Tokyo and Kobe, 1991.
[130] ——, "Unsupervised neural net classification of power system static security states," *Int. J. Electrical Power and Energy Syst.*, vol. 14, no. 2–3, pp. 233–242, Apr.–June 1992.
[131] ——, "Power system static security assessment using the Kohonen neural network classifier," *IEEE Trans. Power Syst.*, vol. 7, pp. 865–872, May 1992.
[132] J. O'Brien and C. Reeves, "Comparison of neural network paradigms for condition monitoring," in *Proc. 5th Int. Congress on Condition Monitoring and Diagnostic Engineering Management*, R. B. K. N. Rao and G. J. Trmal, Eds. Bristol, UK: Univ. W. England, 1993, pp. 395–400.
[133] E. Oja, *Subspace Methods of Pattern Recognition*. Letchworth, U.K.: RSP/Wiley, 1983.
[134] ——, "Self-organizing maps and computer vision," in *Neural Networks for Perception: Human and Machine Perception*, vol. 1, H. Wechsler, Ed. Boston: Academic, 1992, pp. 368–385.
[135] E. Oja and J. Lampinen, "Unsupervised learning for feature extraction," in *Computational Intelligence: Imitating Life*, J. M. Zurada, R. J. Marks II, and C. J. Robinson, Eds. New York: IEEE Press, pp. 13–22, 1994.
[136] M. Peng, C. L. Nikias, and J. G. Proakis, "Adaptive equalization for PAM and QAM signals with neural networks," *Conf. Record of the 25th Asilomar Conf. on Signals, Syst. and Computers*, vol. 1. New York: IEEE Comput. Soc., 1991, pp. 496–500.
[137] G. Poggi and E. Sasso, "Codebook ordering techniques for address-predictive VQ," in *Proc. ICASSP-93, Int. Conf. on Acoust., Speech Signal Process.*, Piscataway, NJ, 1993, vol. 5, pp. 586–589.
[138] J. G. Proakis, "Advances in equalization for intersymbol interference" in *Advances in Communications Systems Theory and Applications*, vol. 4, A. J. Viterbi, Ed. New York: Academic, 1975.
[139] ——, *Digital Communications*, 2nd ed. New York: McGraw-Hill, 1989.
[140] L. Raffo, D. D. Caviglia, and G. M. Bisio, "Neural clustering algorithms for classification and pre-placement of VLSI cells," in *Proc. COMPEURO'92*, The Hague, Netherlands, May 4–8, 1992, pp. 556–561.
[141] K. Raivio, O. Simula, and J. Henriksson, "Improving decision feedback equaliser performance using neural networks," *Electron. Lett.*, vol. 27, no. 23, pp. 2151–2153, 1991.
[142] K. Raivio and T. Kohonen, "Detection of nonlinearly distorted and two-path propagated signals using SOM-based equalizers," in *Proc. ICANN'94, Int. Conf. on Artificial Neur. Networks*, Springer, London, 1994, vol. 2, pp. 1037–1040.
[143] K. Raivio, J. Henriksson, and O. Simula, "Neural detection of QAM modulation in the presence of interference," in *Proc. ICNN'95, Int. Conf. on Neur. Networks*, 1995.
[144] E. A. Riskin, L. E. Atlas, and S.-R. Lay, "Ordered neural maps and their applications to data compression," in *Proc. Workshop on Neur. Networks for Signal Processing*, B. H. Juang, S. Y. Kung, and C. A. Kamm, Eds., Piscataway, NJ, 1991, pp. 543–551.
[145] H. Ritter, T. Martinetz, and K. Schulten, *Neural Computation and Self-Organizing Maps: An Introduction*. Reading, MA: Addison-Wesley, 1992.
[146] H. Ritter, "Parametrized self-organizing maps," in *Proc. ICANN'93 Int. Conf. on Artificial Neur. Networks*, S. Gielen and B. Kappen, Eds. London, UK: Springer, 1993, pp. 568–575.
[147] ——, "Parametrized self-organizing maps for vision learning tasks," in *Proc. ICANN'94, Int. Conf. on Artificial Neur. Networks*, M. Marinaro and P. G. Morasso, Eds. London: Springer, 1994, vol. 2, pp. 803–810.
[148] H. Robbins and S. Monro, "A stochastic approximation method," *Ann. Math. Statist.*, vol. 22, pp. 400–407, 1951.
[149] R. Sadananda and A. Shestra, "Topological maps for VLSI placement," in *Proc. IJCNN-93-Nagoya, Int. Joint Conf. on Neur. Networks*, Piscataway, NJ, 1993, vol. 2, pp. 1955–1958.
[150] J. W. Sammon Jr., "A nonlinear mapping for data structure analysis," *IEEE Trans. Computers*, vol. C-18, pp. 401–409, 1969.
[151] L. J. Scaglione, "Neural network application to particle impact noise detection," in *Proc. ICNN'94, Int. Conf. on Neur. Networks*, Piscataway, NJ, 1994, pp. 3415–3419.
[152] A. Schnettler and V. Tryba, "Artificial self-organizing neural network for partial discharge source recognition," *Archiv für Elektrotechnik*, vol. 76, pp. 149–154, 1993.
[153] A. Schnettler and M. Kurrat, "Partial discharge diagnosis using an artificial neural network," in *Proc. 8th Int. Symp. on High Voltage Engineering*, Yokohama, 1993, pp. 57–60.
[154] T. Shen, J.-R. Gan, and L.-S. Yao, "Application of self-organization neural network in VLSI placement," *Chinese J. Computers*, vol. 15, no. 9, pp. 648–654, 1992.
[155] ——, "Application of fuzzy neural computing in circuit partitioning," *Chinese J. Computers*, vol. 15, no. 9, pp. 641–647, 1992.
[156] ——, "A neural network approach to cell placement," *Acta Electronica Sinica*, vol. 20, no. 10, pp. 100–105, Oct. 1992.
[157] O. Simula and A. Visa, "Self-organizing feature maps in texture classification and segmentation," in *Artificial Neural Networks*, vol. 2, I. Aleksander and J. Taylor, Eds. Amsterdam: North-Holland, 1992, pp. 1621–1628.
[158] O. Simula and J. Kangas, "Process monitoring and visualization using self-organizing maps" in *Neural Networks for Chemical Engineers*, A. B. Bulsari, Ed. New York: Elsevier, 1995, ch. 14.
[159] P. H. Skinnemoen, "Robust communication with modulation organized vector quantization," Ph.D. dissertation, Universitet i Trondheim, Norway, 1994.
[160] P. J. C. Skitt, M. A. Javed, S. A. Sanders, and A. M. Higginson, "Process monitoring using auto-associative, feed-forward artificial neural networks," *J. Intell. Manuf.*, vol. 4, no. 1, pp. 79–94, Feb. 1993.
[161] T. Sorsa, H. N. Koivo, and H. Koivisto, "Neural networks in process fault diagnosis," *IEEE Trans. Syst., Man, Cybern.*, vol. 21, no. 4, pp. 815–825, 1991.
[162] T. Sorsa, H. N. Koivo, and R. Korhonen, "Application of neural network in the detection of breaks in a paper machine," in *IFAC Symp. on On-Line Fault Detection and Supervision in the Chemical Process Industries*, preprints, Newark, DE, Apr. 1992, pp. 162–167.
[163] T. Sorsa and H. N. Koivo, "Application of artificial neural networks in process fault diagnosis," *Automatica*, vol. 29, no. 4, pp. 843–849, 1993.
[164] M. Takahashi, K. Kyuma, and E. Funada, "10000 cell placement optimization using a self-organizing map," in *Proc. IJCNN-93-Nagoya, Int. Joint Conf. on Neur. Networks*, Piscataway, NJ, 1993, vol. 3, pp. 2417–2420.
[165] H. Tang and O. Simula, "The optimal utilization of multi-service SCP," in *Proc. IFIP-TCG Working Conf. on Intell. Networks*, Copenhagen, Denmark, 1995, pp. 157–167.
[166] ——, "Neural adaptation for optical traffic shaping in telephone systems," in *Proc. ICNN'95, Int. Conf. on Neur. Networks*, Perth, Australia, 1995.
[167] C. W. Therrien, *Decision, Estimation and Classification*. New York: Wiley, 1989.
[168] K. Torkkola *et al.*, "Status report of the Finnish phonetic typewriter project," in *Artificial Neural Networks*, T. Kohonen, K. Mäkisara, O. Simula, and J. Kangas, Eds. Amsterdam: North-Holland, 1991, vol. 1, pp. 771–776.
[169] W. Trumper, "A neural network as a self-learning controller," *Automatisierungstechnik*, vol. 40, no. 4, pp. 142–147, Apr. 1992.
[170] V. Tryba, S. Metzen, and K. Goser, "Designing basic integrated circuits by self-organizing feature maps," in *Neuro-Nîmes '89. Int. Workshop on Neur. Networks and Their Applications*, Nanterre, France, 1989, pp. 225–235.

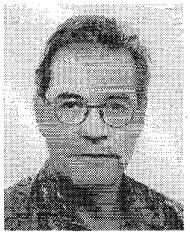
[171] V. Tryba and K. Goser, "Self-organizing feature maps for process control in chemistry," in *Artificial Neural Networks*, T. Kohonen, K. Mäkisara, O. Simula, and J. Kangas, Eds. Amsterdam: North-Holland, 1991, pp. 847–852.
[172] A. Ultsch, "Self organized feature maps for monitoring and knowledge acquisition of a chemical process," in *Proc. ICANN'93, Int. Conf. on Artificial Neur. Networks*, S. Gielen and B. Kappen, Eds. London: Springer, 1993, pp. 864–867.
[173] L. V. Gool, P. Dewaele, and A. Oosterlinck, *Survey Texture Analysis Anno 1983, Computer Vision, Graphics, and Image Processing*, vol. 29, pp. 336–357, 1985.
[174] M. Vapola, O. Simula, T. Kohonen, and P. Meriläinen, "Representation and identification of fault conditions of an anaesthesia system by means of the self-organizing map," in *Proc. ICANN'94 Int. Conf. on Artificial Neur. Networks*, M. Marinaro and P. G. Morasso, Eds. Berlin: Springer-Verlag, 1994, vol. 1, pp. 246–249.
[175] A. Visa, "Identification of stochastic textures with multiresolution features and self-organizing maps," in *Proc. 10th Int. Conf. on Patt. Recognition*, Atlantic City, NJ, June 16–21, 1990, pp. 518–522.
[176] ——, "A texture classifier based on neural network principles," in *Proc. Int. Joint Conf. on Neur. Networks*, San Diego, CA, June 17–21, 1990, vol. 1, pp. 491–496.
[177] A. Visa and A. Langinmaa, "A texture based approach to evaluate solid print quality," in *Proc. IARIGAI*, W. H. Banks, Ed. London: Pentech, 1992.
[178] A. Visa, K. Valkealahti, J. Iivarinen, and O. Simula, "Experiences from operational cloud classifier based on self-organizing map," *SPIE Applicat. of Artificial Neur. Networks V*, 1994, vol. 2243, pp. 484–495.
[179] A. Waibel *et al.*, "Phoneme recognition using time-delay neural networks," *IEEE Trans. Acoust., Speech Signal Process.*, vol. 37, no. 3, pp. 328–339, 1989.
[180] J. Walter and H. Ritter, "Local PSOMs and Chebyshev PSOMs improving the parametrized self-organizing maps," in *Proc. ICANN'95, Int. Conf. on Artificial Neur. Networks*, F. Fogelman–Soulié and P. Gallinari, Eds. Paris: EC2&Cie, 1995, vol. 1, pp. 95–102.
[181] S. Watanabe *et al.*, "Evaluation and selection of variables in pattern recognition," in *Computer and Information Sciences*, vol. 2, J. Tou, Ed. New York: Academic, 1967, pp. 91–122.
[182] F. Wen and Z. Han, "Combined use of Kohonen's model and BP model for the calculation of energy losses in distribution systems," in *3rd Bienn. Symp. on Indust. Electric Power Applicat.*, Ruston, LA, Louisiana Tech. Univ., 1992, pp. 268–277.
[183] J. S. Weszka, C. R. Dyer, and A. Rosenfeld, "A comparative study of texture measures for terrain classification," *IEEE Trans. Syst. Man and Cybern.*, vol. SMC-6, pp. 269–285, Apr. 1976.
[184] G. Whittington and T. Spracklen, "The application of a neural network model to sensor data fusion," in *Proc. SPIE—The Int. Soc. for Opt. Engineering*, vol. 1294, pp. 276–283, 1990.
[185] J.-M. Wu, J.-Y. Lee, Y.-C. Tu, and C.-Y. Liou," "Diagnoses for machine vibrations based on self-organization neural network," in *Proc. IECON'91, Int. Conf. on Industrial Electron., Contr. and Instrumentation*, Piscataway, NJ, 1991, vol. 2, pp. 1506–1510.
[186] L. Xu and E. Oja, "Randomized Hough transform (RHT): Basic mechanisms, algorithms, and complexities," *Computer Vision, Graphics, and Image Processing: Image Understanding*, vol. 57, pp. 131–154, 1993.
[187] T. Yamaguchi, M. Tanabe, J. Murakami, and K. Goto, "An adaptive control with fuzzy associative memory system," *Trans. Inst. Electrical Engineers of Japan, Part C*, vol. 111-C, no. 1, pp. 40–46, Jan. 1991.
[188] T. Yamaguchi, M. Tanabe, and T. Takagi, "Fuzzy associative memory applications to control," in *Artificial Neural Networks*, T. Kohonen, K. Mäkisara, O. Simula, and J. Kangas, Eds. Amsterdam: North-Holland, vol. 2, pp. 1249–1252, 1991.
[189] T. Yamaguchi, M. Tanabe, K. Kuriyama, and T. Mita, "Fuzzy adaptive control with an associative memory system," in *Int. Conf. on Contr. '91 (Conf. Publ. No. 332)*. London: IEE, 1991, vol. 2, pp. 944–949.
[190] M. Yasunaga, M. Asai, K. Shibata, and M. Yamada, "Self-organization capability for eliminating defective neurons in neural network LSI's," *Trans. Inst. of Electron., Information and Commun. Engineers*, vol. J75D-I, no. 11, pp. 1099–1108, Nov. 1992.
[191] I. Yläkoski and A. Visa, "A two-stage classifier for wooden boards," in *Proc. 8th SCIA, Scand. Conf. on Image Anal.*, Tromsø, Norway, NOBIM, 1993, vol. 1, pp. 637–641.
[192] Y. Jilai, G. Zhizhong, and L. Zhuo, "A new fast method for supplying measures to avoid the high voltage mode of electromagnetic voltage transformer," in *Proc. 1st Int. Forum on Applications of Neur. Networks to Power Syst.*, M. A. El-Sharkawi and R. J. Marks II, Eds., Piscataway, NJ, 1991, pp. 293–296.
[193] C.-X. Zhang and D. A. Mlynski, "VLSI-placement with a neural network model," in *Proc. Int. Symp. on Circ. and Syst.*, New Orleans, LA, 1990, pp. 475–478.
[194] C. Zhang and D. A. Mlynski, "Ein neuer VLSI-plazierungsalgorithmus mit neuronalem lernmodell," *GME Fachbericht*, vol. 8, pp. 297–302, 1991.
[195] C. Zhang, A. Vogt, and D. A. Mlynski, "Neuronale plazierungsalgorithmen," *Elektronik*, no. 15, pp. 68–72, 1991.
[196] ——, "Floorplan design using a hierarchical neural learning algorithm," in *Proc. Int. Symp. on Circ. and Syst.*, Singapore, 1991, pp. 2060–2063.
[197] C.-X. Zhang and D. A. Mlynski, "Neural somatotopical mapping for VLSI placement optimization," in *Proc. IJCNN-91 Int. Joint Conf. on Neur. Networks*, Singapore, 1991, pp. 863–868.



Teuvo Kohonen (Fellow, IEEE) received the D.Eng. degree in physics from Helsinki University of Technology, Espoo, Finland, in 1962.

He is a Professor of the Academy of Finland and Professor with the Helsinki University of Technology. His research interests are in associative memories, neural networks, and pattern recognition. He is also an honorary doctor of the University of York, United Kingdom, and Åbo Academy, Finland. He is the author of five textbooks, including *Self-Organization and Associative Memory* (Springer, 1984) and *Self-Organizing Maps* (Springer, 1995).

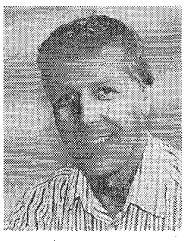
Dr. Kohonen received the IEEE Neural Networks Council 1991 Pioneer Award and the 1995 IEEE Signal Processing Society Technical Achievement Award. He is a member of the Academia Scientarium et Artium Europaea, Académie Européene des Sciences, des Arts et des Lettres, the Finnish Academy of Sciences, and the Finnish Academy of Engineering Sciences. He served as President of the European Neural Network Society and was the first Vice Chairman of the International Association for Pattern Recognition.



Erkki Oja (Senior Member, IEEE) received the Dr.Tech. degree (with distinction) from the Helsinki University of Technology, Espoo, Finland, in 1977.

He is a Professor of Computer Science at the Helsinki University of Technology. He is the author of several journal papers and book chapters on pattern recognition, computer vision, and neural computing, as well as the book *Subspace Methods of Pattern Recognition*. His research interests are in the study of subspace, PCA, self-organizing networks, and applying artificial neural networks to computer vison and signal processing. He serves on the editorial boards of *Neural Networks*, *Neural Computation*, *Pattern Recognition Letters*, and IEEE TRANSACTIONS ON NEURAL NETWORKS.

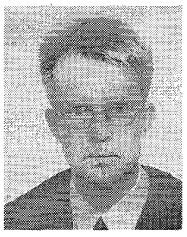
Dr. Oja is a member of the Finnish Academy of Sciences, the Finnish Pattern Recognition Society (past chairman), a fellow and Governing Board member of the International Association of Pattern Recognition, vice president of the European Neural Network Society, and a member of the IEEE Neural Networks Technical Committee.



Olli Simula (Senior Member, IEEE) received the Dr.Tech. degree in computer science from Helsinki University of Technology, Espoo, Finland, in 1979.

He is an Associate Professor of Computer Science at the Helsinki University of Technology. He is the author of several journal and conference papers and book chapters on digital signal processing, image analysis, and neural computing. He edited two conference proceedings on image analysis and artificial neural networks. His research interests include digital signal processing as well as neural networks and their applications in process monitoring, modeling, and analysis.

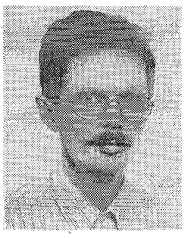
Dr. Simula has served on the governing boards of the Finnish Society and Foundation of Electronics Engineers for several years. He chaired the organizing committees of international conferences, including ISCAS-88 and ICANN-91, the IEEE Finland Section during 1982–1984, and the IEEE Computer Chapter since 1986.



Ari Visa (Senior Member, IEEE) received the Dr.Tech. degree from the Helsinki University of Technology, Espoo, Finland, in 1990.

He is a Docent at the Laboratory of Computer and Information Science, Helsinki University of Technology. Prior to that, he worked on process automation in the Finnish paper and sawmill industry. He is the author of a number of journal and conference papers on image processing, image analysis, pattern recognition, and neural computing. His current research interests are in computational intelligence and image analysis.

Dr. Visa is a member of the SPIE and the Finnish Pattern Recognition Society (past chairman). He served on the organizing committees of a number of conferences.



Jari Kangas received the M.S. and Dr.Eng. degrees in computer science from the Helsinki University of Technology, Espoo, Finland, in 1988 and 1994, respectively.

He is a Junior Research Fellow at the Academy of Finland. His resarch interests are applications of the self-organizing map (SOM) model in various data analysis and data compression tasks, especially the development of error-tolerant vector quantization and image compression by vector quantization, utilizing the special properties of the SOM model. He has authored several technical articles and conference papers in the areas of speech signal analysis and image processing.