appeared in: Neural Networks, Vol.9, No.1, pp.99-120, (1996)

Sensor Encoding Using Lateral Inhibited, Self-organized Cellular Neural Networks

Rüdiger W. Brause[1]

J.W. Goethe-University, Frankfurt (FRG)

Abstract

The paper focuses on the division of the sensor field into subsets of sensor events and proposes the linear transformation with the smallest achievable error for reproduction: the transform coding approach using the principal component analysis (PCA).

For the implementation of the PCA, this paper introduces a new symmetrical, lateral inhibited neural network model, proposes an objective function for it and deduces the corresponding learning rules. The necessary conditions for the learning rate and the inhibition parameter for balancing the crosscorrelations vs. the autocorrelations are computed. The simulation reveals that an increasing inhibition can speed up the convergence process in the beginning slightly.

In the remaining paper, the application of the network in picture encoding is discussed. Here, the use of non-completely connected networks for the self-organized formation of templates in cellular neural networks is shown. It turns out that the self-organizing Kohonen map is just the non-linear, first order approximation of a general self-organizing scheme. Hereby, the classical transform picture coding is changed to a parallel, local model of linear transformation by locally changing sets of self-organized eigenvector projections with overlapping input receptive fields. This approach favors an effective, cheap implementation of sensor encoding directly on the sensor chip.

Keywords Transform coding, Principal component analysis, Lateral inhibited network, Cellular neural network, Kohonen map, Self-organized eigenvector jets.

[1] Author address is PD Dr. R. Brause
J. W. Goethe-Universität, FB15 Inst.f.Informatik, ASA
D- 60054 Frankfurt, Brause@Informatik.Uni-Frankfurt.de

1. INTRODUCTION

The encoding of sensor information is not only an interesting, but also a very important subject. Results are used in picture, speech and music encoding and compression which is needed in applications like telecommunication, satellite data transmission, environmental and geographical image bases, music compression systems, high-resolution television transmission and storage and, therefore, in multi-media data bases. It is also an important subject in the preprocessing for speech recognition or in tactile and position sensing for robot control.

Classically, the sensor signals are seen as locally and time-varying features decomposed by a new feature set which is better usable, e.g. by the Fourier coefficients of a Fourier transformation. This approach is often used for instance in picture processing (Jain, 1989) or in the preprocessing stages of speech-recognition systems, e.g. Kohonen, (1988a). Very often, the Fourier transformation can be done very efficiently in parallel in a local region by small, parallel processing units. In this case, as localized Fourier transform the *Gabor* transformation (see e.g. Daugman, 1988) is considered. The set of the absolute values of the (complex) coefficients of a local linear decomposition can be termed a *jet* ; in the case of Gabor functions it is a *Gabor jet* (Buhmann et al., 1989).

The Gabor decomposition functions have some similarity with the receptive fields found in the visual cortex of cat (Jones & Palmer, 1987; Daugman, 1988). Providing the ON/OFF centre receptive fields can also lead to self-organized, orientation-sensitive receptive fields (Barrow, 1987). Each Gabor jet can be regarded as a topographic organization of the visual input encoding and can be identified with a hypercolumn found in the visual cortex (Okajima, 1986).

Nevertheless, this approach has some flaws: the coefficients of such a decomposition can be correlated; by their interdependence they are not optimally coded and contain unnecessary redundancy. Therefore, let us introduce in this paper another approach by the means of another decomposition or other sensor primitives.

1.1 Sensor Primitives

Assume that we have a set of sensor events $\{\mathbf{x}^{(i)}\}$ where $\mathbf{x}^{(i)} = (x_1^{(i)},\ldots,x_n^{(i)})$ denotes a tuple of real sensor data which characterizes event *i*. Some events occur more often than others. If we have one input pattern $\mathbf{x}^{(k)} = \mathbf{w}$ which occurs mostly we note that the situation $\{x_1 = x_1^{(k)}, \ldots. , x_n = x_n^{(k)}\}$ is often observed; the data values are no more random but there exists now a *correlation* between the input and the pattern $\mathbf{w}$. Therefore, to reduce the input information we can construct a device which reacts especially to this input pattern and describe the space of input tuples in terms of a prototype pattern and residuals. This device is a correlation detector, shown in Fig. 1.

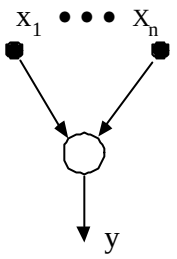

Fig. 1 *A correlation detection device. The device has its highest output when the correlation between the input signal pattern and the weight pattern coincidents the most*

The output y of the device can give a measure of the correlation or similarity of the input $\mathbf{x}$ and the prototype pattern $\mathbf{w}$ such that

$$y = \sum_j x_j w_j = \mathbf{x}^T \mathbf{w} \qquad (1.1)$$

with ^{T} denoting the vector transpose. A performance criterion for such a correlation device could be that it detects the most often used input by the expected correlation with $\mathbf{w}$

$$\max_{\mathbf{w}} \langle y \rangle_{opt} = \max_{\mathbf{w}} \langle y \rangle$$

using the expectation operator $\langle\ \rangle$. Also a strong negative correlation is an indication for an event, because a strong negative output reveals also the appearance of $\mathbf{x}$ for anti-correlated patterns $\mathbf{w}$ which respond maximally when $\mathbf{x}$ is not present. Therefore, we change the goal to a sign-independent version

$$\max_{\mathbf{w}} \langle y^2 \rangle_{opt} = \max_{\mathbf{w}} \langle y^2 \rangle = \max_{\mathbf{w}} \langle \mathbf{w}^T \mathbf{x}\mathbf{x}^T \mathbf{w} \rangle = \max_{\mathbf{w}} \mathbf{w}^T \mathbf{C}_{xx} \mathbf{w} \qquad (1.2)$$

with the autocorrelation matrix $\mathbf{C}_{xx} = \langle \mathbf{x}\mathbf{x}^T \rangle$.

This unit is responsible for the most often used correlation. Additional device units can be used to recognize less frequently occurring input events by other prototypes $\mathbf{w}_i$. In order to distinguish the input events maximally, the response y_i of the prototype $\mathbf{w}_i$ should be not correlated to another one. Thus, a network of correlation detection devices which use eqn (1.2) can be seen as a network which describes the input by terms of often occurring uncorrelated events or signal primitives.

What is the solution for eqn(1.2)? Certainly, for unrestricted $\mathbf{w}$ there are trivial solutions like $w_j = 0$ or $w_j = \infty$. Since several, parallel working devices of the linear type implement a linear transformation, we might choose as device function restriction that this transformation should be neutral, i.e., the volume spanned by the

new base vectors should be the same as by the old ones. This is accomplished by the demand $\det(\mathbf{w}_1,\ldots,\mathbf{w}_m) = 1$ which in turn results by the (more restrictive) demand for an orthogonal base of $|\mathbf{w}_i| = 1$ because then we get with $\mathbf{W} = (\mathbf{w}_i,\ldots,\mathbf{w}_m)^T$ and $\mathbf{W}\mathbf{W}^T = \mathbf{I}$ the proportion $1 = \det(\mathbf{I}) = \det(\mathbf{W}\mathbf{W}^T) = \det(\mathbf{W})\cdot\det(\mathbf{W}^T) = \det^2(\mathbf{W})$.

It can be shown that for non-zero, constant-length $\mathbf{w}$ the objective function $R(\mathbf{w}) = \langle y^2_{(\mathbf{w})} \rangle$ takes its extremes at $\mathbf{w} = \mathbf{e}_i$, the eigenvectors of $\mathbf{C}_{xx}$. When we have only different, distinct eigenvalues, there is one unique maximum for the eigenvector with the maximal eigenvalue and one unique minimum for the eigenvector with the smallest eigenvalue (e.g. Brause, 1992b).

1.2 The Minimal Mean Square Error Of The Sensor Primitives

If we use the same number m of devices as there are input lines, the $m = n$ output values y_i are just the projection of the input $\mathbf{x}$ on the vectors $\mathbf{w}_i$ or the coordinates of $\mathbf{x}$ in a new base $\{\mathbf{w}_i\}$. When the $\mathbf{w}_i$ are linearly independent and complete, we do not loose information and a complete, errorless reconstruction of the input $\mathbf{x}$ by $\mathbf{y} = (y_1,\ldots,y_m)$ is possible.

However, if we use $m<n$, i.e., less devices than input lines, we will make an error attempting to reconstruct $\mathbf{x}$ from $\mathbf{y}$. To reduce this error, we will first choose as a description base $\{\mathbf{w}_i\}$ the m most important (most frequent) input patterns of eqn (1.2). For many purposes the necessary processing of sensor input signals is realized by using a system which implements the maximization of the transinformation from the input to the output of the system. For deterministic systems, this corresponds to the maximization of the output entropy (maximum entropy principle). In pattern recognition theory, it is well known that for Gaussian distributed sources processed by linear systems this corresponds to the *minimization of the mean square error* of the output (Tou & Gonzalez, 1974).

The decomposition of the input into eigenvector components which minimizes the mean squared error criterion is also called a "discrete Karhunen-Loéve transformation (KLT)", "Hotelling transformation", "principal component analysis (PCA)" or "eigenvector decomposition".

For linear systems, it is well known that the mean square error is minimized by selecting only those base vectors (eigenvectors) with the biggest eigenvalues (Kramer & Matthews, 1956; Fukunaga, 1972). This corresponds in our notation of section 1.1 to the most frequent events. Neglecting the ones with the smallest eigenvalues results in the smallest reconstruction error of the encoded input. The squared error for using only $m < n$ components by three different transformations is shown in figure 2 for a homogeneous random field, i.e., a picture containing random pixel data according to a translation-invariant autocorrelation function.

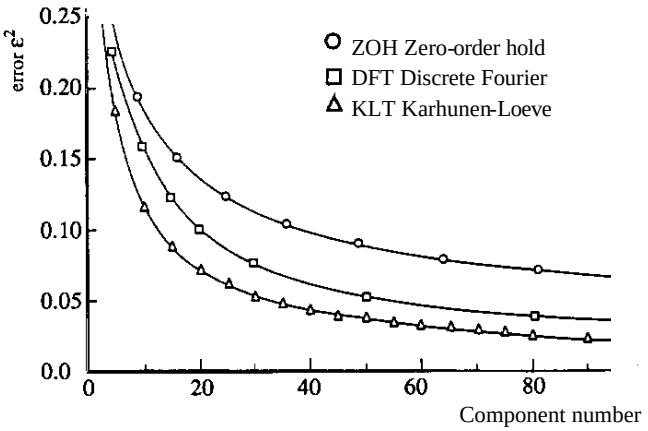

Fig. 2 *The squared error for neglecting high order components, plotted as function of the highest component index still used, and compared for three different encoding methods (after Habibi & Wintz,1971)*

Since we consider only pictures of discrete pixels with integer-valued indices both for **x** and **y**, we use only discrete transformations containing finite sums.

Obviously (by definition!), the eigenvector decomposition KLT performs the best and can be considered as an *optimal* transformation for the mean squared error criterion. It should be preferred to all other current linear transformations as the simple Zero-order-hold transformation ZOH which computes the local average in a pixel region, the Hadamard transformation which use special binary functions, the commonly used discrete Fourier transformation DFT or its real-valued version, the discrete Cosines transformation DCT which describe the picture by its frequencies (Habibi & Wintz,1971).

The reason why the Fast Fourier Transform (FFT) or the Fast Cosine Transform (FCT) and not the KLT are often chosen as encoding standard lies mainly in the fact that in sequential implementations, the run time complexity of the KLT is $O(N^6)$ for a picture with N×N pixels whereas for the FTT this is reduced to $O(N^2(\log N)^{2)}$, see (Jayant & Noll, 1984). However, if the base vectors are already known because the input statistics are stable, the complexity is reduced to the one of a linear transform which can be implemented in parallel hardware as shown in this paper. This makes the KLT approach attractive again as a good candidate for encoding purposes.

1.3 PCA decomposition of sensor signals

The decomposition of sensor signals (*transform coding*) into eigenvector components leads to the least mean square error for the reconstruction of the original signal by the components y_i and the eigenvectors $\mathbf{e}_i$. If we treat a picture as a signal, all discrete n = N×M pixels of the picture can be arranged in one input vector of *n* components. Thus, also the eigenvectors of the corresponding autocorrelation matrix have *n* components and can be rearranged back into picture form: they are the *basis images* of the decomposition and called *eigen images* (Jayant & Noll, 1984) and represent the pattern primitives we have looked for in section 1.1.

The correlations in pictures decrease rapidly with increasing distance. Wintz (1972) reports that for image reproduction it suffices to consider correlations only 4-5 pixels wide. Therefore, instead of including all correlations on N×M pixels we divide the picture into *K* subpictures and describe the whole picture by *K* sets of eigenvectors with length *n* = N×M/K, see figure 3.

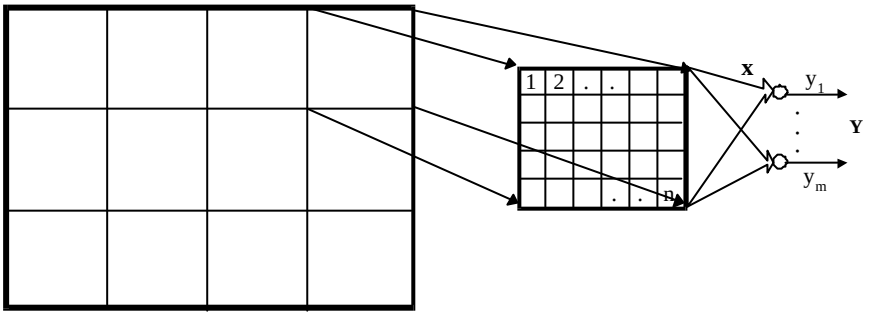

Fig. 3 *The picture decomposition by parallel processing systems. The picture is devided in subpictures. The sensor elements (pixels) of each subpicture can serve as the input for a neural network, processing the whole picture in parallel. Conventionally, all subpictures are processed in sequential order, leading to a linearization in run time complexity of the sequential algorithm used for each subpicture. Since the long-range correlations of the pixels are neglected by the subpicture approach, it yields a certain (neglectable) error.*

This approach [which was first proposed by Habibi & Wintz, 1971] breaks the encoding process into parallel activities for *K* independent working processing systems. The price we pay is an small error, depending on our subpicture size and our pixel correlation statistics. By experience, a subpicture size of 8x8 pixels can be considered to be sufficient (Wintz, 1972).

Now, let us identify each one of the parallel processing systems by a subnet of *m* neurons. The parallel, distributed encoding of the whole picture is done by a neural network where each subpicture of *n* pixels is coded by a local transformation process into *m* components by a subnetwork of *m* neurons.

1.4 The transform coding concept

The classical transform encoding process consists of two stages: a linear transformation, which for instance is implemented in the JPEG and MPEG standard video encoding by a discrete cosine transform (JPEG-9-R6, 1991; MPEG 91, 1991), and a vector quantization stage. Both stages contain non-linear operations and reduce the data stream; the linear transformation projects components on constant values which results in a dimension reduction (non-zero kernel) and the vector quantization maps all data of the neighborhood to only several class prototypes. The image coding and decoding is illustrated in Fig. 4.

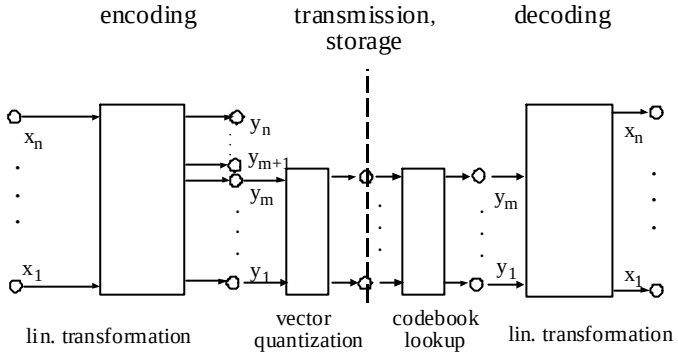

Fig. 4 *The image coding and decoding schema of* transform coding. *After a linear transformation and a quantization stage, the compressed signal is stored or transmitted. At the receiver or display device it is restored in the inverse order. The compression is obtained by neglecting* n-m *components of the linear decomposition and the additional quantization of the remaining ones.*

This concept can be implemented by integrating the neural network directly on a VLSI chip (e.g. a CCD chip) in parallel to the light-sensitive cells. The neurons will then directly learn the eigenvectors by the input signal statistics, see e.g. Brause (1994). Nevertheless, the output can be sequential as the usual video signal. On the receiver side, the reconstruction of the picture signal can be done directly on the screen, e.g. a LCD. Since we can get the values for the weights by training a simulated system like this with pictures of the desired statistics or use directly analytical solutions, we can implement the weights on the sender and on the receiver side of the system as pure ROM solutions without complicated learning mechanism. Here, the neural network model

which will be introduced in section 2 just serves a simulation tool for the training phase and is abandoned for the activity phase, the usage of the implemented system.

In this paper we present a neural model for the first encoding stage, although the second stage, the vector quantization, can also be modeled by the use of non-linear versions of this network, e.g. the Kohonen map (Kohonen, 1982).

1.5 PCA Encoding by Neural Networks

There already exist several neural nets for the implementation of an eigenvector decomposition. Let us start with Oja's statement (Oja, 1982) that a linear, formal neuron using Hebb's learning rule and restricted weights will learn the eigenvector of the expected autocorrelation matrix $\mathbf{C}_{xx}$ of the input patterns $\mathbf{x}$ with the biggest eigenvalue λ_{max} (which was partially anticipated by Amari, 1972):

$$\mathbf{w} \rightarrow \mathbf{e}^k \text{ with } \lambda_k = \max_i \lambda_i$$

and $$\mathbf{C}_{xx}\mathbf{e}^i = \lambda_i \mathbf{e}^i \qquad \mathbf{C}_{xx} = \langle \mathbf{x}\mathbf{x}^T \rangle \ .$$

Since then several network architectures were proposed for a partial or complete eigenvector decomposition. Basically, they consist of two categories: networks which learns the eigenvectors sequentially ("asymmetric networks") which are based on the sequential Gram-Schmidt orthogonalization mechanism, and networks which learn them in parallel ("symmetric networks") and do not predetermine an order of the eigenvectors. The approaches use linear neurons, where each neural weight vector converges to one eigenvector.

Examples of the former architectures are the Sanger (1989) decomposition network, and the lateral inhibition network of Rubner and Tavan (1989). They use as a basic building block the linear correlation neuron which learns the input weights by a Hebb-rule, restricting the weights $w_1,\ldots,w_n$. As Oja (1982) showed, this learning rule let the weight vector of the neuron converge to the eigenvector of the expected autocorrelation matrix.

The learning rule for one neuron can be generalized, yielding a network where the input is inhibited simultaneously by the projections of the input to all weight vectors. This corresponds to the latter, symmetric network approach. All those symmetric networks, as the Oja (1989) subspace network, the Williams (1985) subspace learning and the lateral inhibition network of Földiák (1989), which is a version of Kohonen and Oja (1976) orthogonalizing filter, have the property that they provide the convergence of the weight vectors only to the subspace of the eigenvectors, not necessarily to the eigenvectors themselves.

In this paper a new fully symmetrical network for eigenvector decomposition, constructed by an objective function and, different to the networks needing weight feedback or interneurons (Oja 1992; Plumbley 1993) and implemented by a biological plausible and easily realizable network mechanism is presented. Contrary to the opinion of Hornik and Kuan (1992), who are not in favor of symmetric local PCA algorithms because they do not give rise to asymptotically stable desired equilibria and have generally a slower convergence than their asymmetric counterparts, we will introduce a new symmetric model in this section which is not covered by their general convergence analysis of the PCA models mentioned above and which will enable us to observe the self-organized local eigenvector decomposition developments reported in Section 4.

2. THE SYMMETRIC BASE MODEL

Now, let us describe in this section a new symmetric learning model which results in weight vectors implementing a PCA. This was first introduced by Brause (1993a,b); a similar but not identical model was independently developed by Freisleben (1993) and Leen (1991) .

2.1 The Activity Model

Let us assume in a first step that we have *m* neurons which are laterally interconnected as shown in Fig. 5.

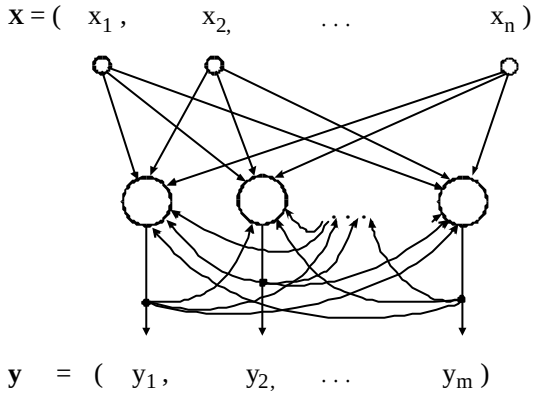

Fig. 5 *The symmetric, laterally interconnected network model. Input and output are symmetrically distributed to all neurons which are linear ones. All weights are randomly initialized; there is no preassignment of the eigenvector index to a weight index. At the end of the convergence, the lateral inhibition weights which represent the expected cross correlation between the neural outputs become zero*

Each neuron *i* has a randomly chosen prototype ("weight vector") $\mathbf{w}_i$. After we presented one input pattern $\mathbf{x}$ in parallel with each neuron of the linear system, the output of neuron *i* will result in

$$y_i = \mathbf{w}_i^T\mathbf{x} + T_i \qquad T_i = \sum_{j\neq i} u_{ij}y_j \tag{2.1}$$

where T_i denotes the influence by the lateral connections which are weighted by the lateral weights u_{ij}. The input can be assumed to be zero-mean, i.e., $\langle \mathbf{x} \rangle = 0$. If this is not the case, it can be made so by introducing a special threshold weight learned with an anti-Hebb-rule, see Appendix C.

Although the model is quite linear, we have reactions for random input and weights due to the feedback lines which are difficult to analyze. Nevertheless, for the prediction of the system behaviors the analysis of the expected equilibrium states of the system is sufficient.

Let us assume that after an input pattern has been presented the system activity stabilizes [see for example (Kohonen, 1976)]. According to theorem 3 of Hirsch (1989), the system (2.1) is globally asymptotically stable for any input for symmetrical $u_{ij} = u_{ji}$ when $u_{ii} + \sum_{j\neq i} |u_{ij}| < 1$ and there is no significant feedback delay (Marcus et al., 1991). In the case where the condition is not met initially, this can be assured by a normalization and small decrement $u_{ij} \rightarrow (u_{ij}/\Sigma_j|u_{ij}|) - \varepsilon$ even for $u_{ii} \geq 0$. Therefore, with $\varepsilon > 0$ the expression $u_{ii} + \sum_{j\neq i} |u_{ij}|$ becomes

$$\frac{u_{ii}}{\sum_k |u_{ik}|} - \varepsilon + \sum_{i\neq j} \frac{u_{ij}}{\sum_k |u_{ik}|} - \varepsilon = \left(\sum_j \frac{|u_{ij}|}{\sum_k |u_{ik}|}\right) - m\varepsilon + \frac{u_{ii}}{\sum_k |u_{ik}|} - \frac{u_{ii}}{\sum_k |u_{ik}|} \leq 1 - m\varepsilon$$

and the stability condition of Hirsch is satisfied.

Then the output for neuron *i* becomes with Eqn (2.1) and the definition $u_{ii} = 1$

$$y_i = \mathbf{w}_i^T\mathbf{x} + \sum_{i\neq j} u_{ij}y_j = \mathbf{w}_i^T\mathbf{x} + \mathbf{u}_i^T\mathbf{y} - y_i$$

and the output vector of all neurons becomes

$$2\mathbf{y} = \mathbf{W}\mathbf{x} + \mathbf{U}\mathbf{y} \quad \text{or} \quad (2\mathbf{I}-\mathbf{U})\mathbf{y} = \mathbf{W}\mathbf{x}$$

with the identity matrix $\mathbf{I}$.

Thus, the system output

$$\mathbf{y} = (2\mathbf{I}-\mathbf{U})^{-1}\mathbf{W}\,\mathbf{x} = \mathbf{B}\,\mathbf{x} \qquad \mathbf{B} = (2\mathbf{I}-\mathbf{U})^{-1}\mathbf{W} \tag{2.2}$$

depends again linearly on the input.

2.2 The Learning of the Weights

The learning rule for the weights $\mathbf{b}_i$ of $\mathbf{B} = (\mathbf{b}_i)$ is determined by the following three conditions, introduced in Section 1.1:

- The responses to different pattern prototypes should not be correlated

$$\langle y_i y_j \rangle = \langle y_i \rangle \langle y_j \rangle = 0 \qquad \forall\, i \neq j \tag{2.3}$$

- The response to the pattern prototypes should be maximal

$$\sum_i \langle y_i^2 \rangle = \max. \tag{2.4}$$

- The change in the feature base should be neutral (no scaling)

$$\det(\mathbf{B}) = 1, \text{ e.g. } |\mathbf{b}_i| = 1 \text{ suffices for an orthogonal base} \tag{2.5}$$

The first two conditions (2.3) and (2.4) can be modeled by the minimum of deterministic objective function

$$R(\mathbf{b}_1,\ldots,\mathbf{b}_m) = {}^1\!/_4\, \beta \sum_i \sum_{j \neq i} (\langle y_i y_j \rangle)^2 - {}^1\!/_2 \sum_i \langle y_i^2 \rangle = R_1 + R_2 \,. \tag{2.6}$$

The first term R_1 ensures that the cross-correlation (2.3) is always counted positive. This results in a minimum of R(.) where by R_1 , the squared cross-correlation (2.3), becomes zero and $-R_2$, the sum of all variances, becomes maximal. Since the extremes of the objective function, even scaled by an arbitrary factor, remain the same, the factor β denotes only the *relative* influence of the cross-correlation with respect to the autocorrelation influence.

The third condition (2.5) has to be additionally assured during the learning process. This condition could also be integrated into the objective function by a proper term, see Freisleben (1993) and Leen (1991). It was shown for one neuron (Chauvin, 1989) that this kind of risk function with implicit weight normalization yields the eigenvectors as solutions. The approach of explicit normalization, i.e., using (2.5) when computing the unique maximum (or minimum) of the objective function by a gradient ascend (or descend), in general yields the same solutions for one neuron (Brause, 1992b).

For several neurons, the situation is more complex. In Appendix A it is shown that the objective function $R(\mathbf{b})$ takes its extremes when the $\mathbf{b}_i$, the rows of the matrix $\mathbf{B}$, are a subset of the eigenvectors of the autocorrelation matrix $\mathbf{C}_{xx} = \langle \mathbf{x}\mathbf{x}^T \rangle$. Since $\mathbf{C}_{xx}$ is symmetric and real, the eigenvalues λ_i are real and the eigenvectors form an orthogonal base system. Here, the cross-correlations

$$\langle y_i y_j \rangle = \mathbf{b}_i^T \langle \mathbf{x}\mathbf{x}^T \rangle \mathbf{b}_j = \mathbf{b}_i^T \mathbf{C}_{xx} \mathbf{b}_j = \mathbf{b}_i^T \lambda_j \mathbf{b}_j = 0 \qquad \forall\, i \neq j$$

become zero, and by the assignment

$$u_{ij} = -\langle y_i y_j \rangle \qquad \forall\, i \neq j \quad \textit{lateral inhibition weights} \tag{2.7}$$

we get as a result of the learning process

$$\mathbf{U} = \mathbf{I} \tag{2.8a}$$

and

$$\mathbf{B} = (2\mathbf{I}-\mathbf{U})^{-1}\mathbf{W} = (2\mathbf{I}-\mathbf{I})^{-1}\mathbf{W} = \mathbf{W} \tag{2.8b}$$

The minimum of the objective function (2.6) is reached when the weight vectors become the eigenvectors of the autocorrelation matrix $\mathbf{C}_{xx}$; the lateral inhibition weights become zero. To learn the weight vectors $\mathbf{b}_i$, a gradient descend may be used. Nevertheless, with (2.2) this leads to complicated expressions for $\mathbf{w}_i$ and u_{ij}. Instead, with (2.8) we can conclude that a simplified learning rule for $\mathbf{w}_i$ which ensures the convergence to the eigenvectors of $\mathbf{C}_{xx}$ will also reach the goal.

For this purpose, we change our neural modeling. For the learning of $\mathbf{W}$, we replace our complicated model by a much simpler one which will give the same results. Different to the activity model of eqn (2.1), for learning we consider a net of simple, linear neurons which are not coupled in the activity phase, i.e., $y_i = \mathbf{w}_i^T\mathbf{x}$, but only in the learning phase. For the objective function (2.6) which now depends only on $\mathbf{w}_i$ the minimum can be approximated by a gradient search for the weight vectors $\mathbf{w}_i$ directly, assuring conditions (2.3), (2.4), (2.5) by the usage of the objective function (2.6) for $\mathbf{b} = \mathbf{w}$. The learning rule for the lateral weights which has to implement the demand of eqn (2.7) is split apart and is treated separately.

Thus, the (t+1)-th iteration step for the input weights is

$$\tilde{\mathbf{w}}_i(t+1) = \mathbf{w}_i(t) - \gamma(t)\,\nabla_w R(\mathbf{w}_i) \tag{2.9}$$

$$\mathbf{w}_i(t+1) = \tilde{\mathbf{w}}_i(t+1)/|\tilde{\mathbf{w}}_i(t+1)| \qquad \textit{normalization} \tag{2.10}$$

denoting the gradient by the Nabla-operator $\nabla_w = (\partial/\partial w_1, \ldots, \partial/\partial w_n)^T$ and a proportional constant by γ which generally depends on the iteration (time) step t.

The gradient of R(.) is computed in Appendix A, eqn (A.2). Substituting (A.2) into the deterministic leaming rule (2.9) we get

$$\tilde{\mathbf{w}}_i(t+1) = \mathbf{w}_i(t) - \gamma(t)\,(\beta \sum_{j\neq i} \langle y_i y_j \rangle \langle \mathbf{x} y_j \rangle - \langle \mathbf{x} y_i \rangle)$$

This becomes with definition (2.7)

$$\tilde{\mathbf{w}}_i(t+1) = \mathbf{w}_i(t) + \gamma(t)\langle \mathbf{x}\,(y_i + \beta\Sigma_{j\neq i}\, u_{ij}y_j)\rangle = \mathbf{w}_i(t) + \gamma(t)\Delta\mathbf{w}_i \qquad (2.11)$$

In our new learning model we do not use the lateral activity, only the correlations. With $y_i = \mathbf{w}_i^T\mathbf{x}$ we get

$$\Delta\mathbf{w}_i = \gamma(t)\left(\langle\mathbf{x}\mathbf{x}^T\rangle\mathbf{w}_i - \beta\Sigma_{j\neq i}\,\langle\mathbf{x}\mathbf{x}^T\rangle\mathbf{w}_j(\mathbf{w}_j^T\langle\mathbf{x}\mathbf{x}^T\rangle\mathbf{w}_i)\right)$$

$$= \gamma(t)\left(\mathbf{C}_{xx}\mathbf{w}_i - \beta\mathbf{C}_{xx}(\Sigma_{j\neq i}\mathbf{w}_j\mathbf{w}_j^T)\mathbf{C}_{xx}\mathbf{w}_i\right) \qquad (2.12)$$

and the stochastic version of (2.11) is

$$\tilde{\mathbf{w}}_i(t+1) = \mathbf{w}_i(t) + \gamma(t)\,\mathbf{x}\,(y_i + \beta\Sigma_{j\neq i}\, u_{ij}y_j) \qquad (2.13)$$

or $$\Delta\mathbf{w}_i = \gamma(t)\,\mathbf{x}\,(y_i + \beta\Sigma_{j\neq i}\, u_{ij}y_j) \qquad (2.14)$$

It should be noted that all learning eqns (2.11)-(2.14) assume that the lateral weights have already perfectly converged to the goal of definition (2.7).

Thus, the lateral weights should be updated separately by a rule which let them become the expected cross-correlation as fast as possible. In Appendix B, a learning rule for learning the expectation value of a stationary random variable v by a parameter r is presented. By replacing literally v by y_iy_j, r by u_{ij} and α by –1 in eqns (B.1) and (B.2) we can apply the proof of appendix B, and (B.1) becomes the learning rule

$$u_{ij}(t) = u_{ij}(t\text{-}1) - \frac{1}{t}(u_{ij}(t\text{-}1) + y_i(t)y_j(t)) \qquad (2.15)$$

using a learning rate of $\gamma(t) = 1/t$. This learning rule gets the average of the random variable $v = y_iy_j$; the learning rate of 1/t weights all observations of v by the same amount, independently of the observed order, see Pfaffelhuber and Damle (1973). But be aware: This is not the quantity we are looking for, because v is not stationary for changing $\mathbf{w}_i$. Therefore, random initial values of the weights can disturb the average for a long period of simulation time. To remove these random values and to accelerate the convergence, we might use instead of the learning rule (2.15) the constant learning rate $\gamma(t) = \gamma$ = const or the temporal floating average of a small number q of observed data

$$u_{ij}(t) = -(1/q)\sum_{k=t-q}^{t-1} y_i(k)y_j(k) \qquad (2.16)$$

which is a kind of weight decay process.

2.3 Stability Conditions and the Learned Fixed Points

The fixed points of the learning system are calculated in Appendix A. Nevertheless, they do not indicate under what conditions (cross-correlation parameter β and learning rate γ) the desired fixed points are stable.

The sequential gradient descend algorithm (2.9) only confirms the existence of fixed points by the monotonic decrease of the quadratic objective function (2.6), because we have for $\mathbf{b} = \mathbf{w}$

$$\frac{dR(t)}{dt} = \sum_i \frac{\partial R(\mathbf{w}_i)}{\partial \mathbf{w}_i} \frac{\partial \mathbf{w}_i}{\partial t} = -\sum_i \frac{\partial R(\mathbf{w}_i)}{\partial \mathbf{w}_i} \gamma \frac{\partial R(\mathbf{w}_i)}{\partial \mathbf{w}_i} = -\gamma \sum_i (\partial R_i)^2 \leq 0 \qquad (2.17)$$

for $\gamma > 0$. The objective function R has a lower bound for *m* neurons of

$$\min(R) = \min(R_1 + R_2), \quad R_1 \geq 0$$

which is limited by the finite value of

$$\min(R_2) = \min\left(-\tfrac{1}{2}\Sigma_i \langle y_i^2 \rangle\right) = -\tfrac{1}{2}\, m\, \max\left(\mathbf{w}_i^T \langle \mathbf{x}\mathbf{x}^T \rangle \mathbf{w}_i\right) = -\tfrac{1}{2}\, m\, \max(\mathbf{w}_i^T \mathbf{C}_{xx} \mathbf{w}_i)$$
$$= -\tfrac{1}{2}\, m\, \max(\mathbf{e}_i^T \mathbf{e}_i \lambda_i) = -\tfrac{1}{2}\, m\, \lambda_{max}\,, \quad \text{with } \lambda_{max} = \max_i \lambda_i$$

for linear systems.

Now we know by relation (2.17) that during the iteration the objective function will diminish until it reaches a minimal value. Finally, in the limit we will have $dR(t)/dt = 0$, i.e., a fixed point of R(t). By (2.17) the objective function satisfies the Ljapunov conditions (see Bronstein & Semendjajew, 1990) for a stable fixed point. Started in a convergence region V_i the iteration will converge to the single fixed point $\mathbf{W}_i^* = (\mathbf{w}_1^*, \ldots, \mathbf{w}_m^*)$ which depends on the starting point $\mathbf{W} = (\mathbf{w}_1, \ldots, \mathbf{w}_m)|_{t=0}$, the input statistics and the parameters β and γ. The boundaries of the domains of attraction, the convergence regions V_i, generally separate different (e.g. global and local) minima of R(t) and are not further evaluated in this paper.

Thus, for the sequential case principally we have shown the convergence of the learning system (2.9) which is a general property of all gradient descend learning rules for bounded objective functions (Bronstein & Semendjajew,1990). Nevertheless, we have to take care if the stable fixed point is a desired one. Let us evaluate the conditions for the parameters β and γ to get the desired eigenvectors.

2.3.1 The Crosscorrelation Factor b

In Appendix A it is proven that all the possible fixed points of the system are at the eigenvectors of the autocorrelation matrix. Note that this means that only the fixed points of the system are eigenvectors, they have not to be necessarily different ones. If we regard the objective function as a kind of "energy" and its values at the fixed points as "energy levels", the question arises whether the neurons ("atoms") might stay all at different energy levels or if they cluster all (or some) at the lowest energy level. What are the conditions which let them drop from higher levels to lower levels?

For discrete time steps $\Delta t = 1$, the convergence condition (2.17) transforms to

$$\Delta R := R(t+1) - R(t) \leq 0 \qquad (2.18)$$

and allows a change in the weights only if the energy decreases. Let us assume that one weight vector $\mathbf{w} = \mathbf{e}_k$ at a fixed point differs from the others. Since they are all the eigenvectors of $\mathbf{C}_{xx}$, a jump of the "atom" from energy level k down to level p augments the energy of the additional correlation term $\Delta R_1 = \beta s_p \lambda_p^2/2$ when we already have s_p weight vectors converged to eigenvector $\mathbf{e}_p$, and decreases the autocorrelation term by $\Delta R_2 = -(\lambda_p - \lambda_k)/2$. Condition (2.18) becomes

$$\Delta R = \Delta R_1 + \Delta R_2 = \beta s_p \lambda_p^2/2 - (\lambda_p - \lambda_k)/2 \leq 0$$

The transition to $s_p = 2$ will be prohibited and the convergence to different eigenvectors will be assured when for $s_p = 1$ the condition

$$\beta \lambda_p^2 - (\lambda_p - \lambda_k) > 0$$

or
$$\beta > (\lambda_p - \lambda_k)/\lambda_p^2 \qquad \forall\ k, p \qquad (2.19)$$

is guaranteed. Let us assume that we have m different eigenvalues. What is the best choice for b to assure different weight vectors ? With $a = \lambda_k$, $x = \lambda_p$ the function $f(x) = (x-a)x^{-2}$ corresponds to eqn (2.19). It takes its maximum at $x = 2a$ with $f(2a) = 1/4a$. The function has its maximal value at $\lambda_k = \lambda_{min}$ and condition (2.19) becomes

$$\beta > (4 \lambda_{min})^{-1} \quad \text{with } \lambda_{min} = \min_{i = 1 \ldots m} \lambda_i \qquad (2.20)$$

as a necessary condition for the convergence under the same b for all neurons to all m desired eigenvectors. It is evident that in the trivial case for $\lambda_{min} = 0$ there is no value of b fitting and therefore no convergence to an eigenvector guaranteed, because in this case every arbitrary, non-orthogonal vector satisfies the characteristic eqn.

2.3.2 The Learning Rate γ

It is evident that with decreasing γ the discrete step dynamics becomes the continuous time dynamics and the convergence is treated by the previous section. Nevertheless, for finite γ this is not true and we have to deal with it separately. Unfortunately, the development of the learning system due to initial and developing mutual correlations is quite complicated. To get some simple conditions for the learning rate which simulations showed to be relevant we limit our analysis to local stability considerations for the nearly converged system.

After step t+1, the new weight vector $\mathbf{w}_i(t+1)$ is, combining eqns (2.9) and (2.10), given by

$$\mathbf{w}_i(t+1) = \frac{\mathbf{w}_i(t) + \Delta\mathbf{w}_i}{|\mathbf{w}_i(t) + \Delta\mathbf{w}_i|} \tag{2.21}$$

Now we might write the weight vector as a linear combination $\mathbf{w}_i(t) = \sum_k a_{ik}(t)\mathbf{e}_k$ of special base vectors, the eigenvectors $\mathbf{e}_k$. In Appendix D, eqn (2.21) is evaluated for a_{ik}, the k-th component of the weight vector and gives us

$$a_{ik}(t+1) = \frac{1}{g}\ \{ a_{ik}(t) + \gamma\lambda_k a_{ik}(t) - \gamma\lambda_k\beta\Sigma_{j\neq i}\, a_{jk}(t)b_{ij}\ \} \tag{D.1}$$

with vector length $g = |\mathbf{w}_i(t) + \Delta\mathbf{w}_i|$

and the weighted cross-correlation coefficients $b_{ij} := \Sigma_k\, a_{jke}(t)\, a_{ik}(t)\lambda_k$ of different neurons.

Since the ratio $a_{ik}(t)/a_{ip}(t)$ of two components *k* and *p* of the weight vector $\mathbf{w}_i(t+1)$ is independent of the length *g* of the weight vector itself, it is interesting to observe the behavior of the weight vector with the changes of the absolute ratio in different eigenvector directions *k* and *p*. If the absolute value of this ratio $|a_{ik}(t)/a_{ip}(t)|$ increases at each time step for every component *p*, we can conclude that weight vector $\mathbf{w}_i$ converges to eigenvector $\mathbf{e}_k$ of the autocorrelation matrix $\mathbf{C}_{xx}$, even when the sign of the ratio itself changes at each iteration step (which can be observed in some simulations).

For the case of just one neuron, the sum $\Sigma_{j\neq i}$ in eqn (D.1) becomes zero and the ratio is

$$\frac{|a_{ik}(t+1)|}{|a_{ip}(t+1)|} = \frac{|a_{ik}(t)(1+\gamma\lambda_k)|}{|a_{ip}(t)(1+\gamma\lambda_p)|}$$

and will increase, if the condition $(1+\gamma\lambda_k) > (1+\gamma\lambda_p)$, i.e., $\lambda_k > \lambda_p$ holds for all other components *p*. Thus, independently of the initial weights, the learning rate γ and the cross-correlation factor *b* the weight vector will converge to the eigenvector with the biggest eigenvalue.

Now, assume that already m weight vectors have converged to the m eigenvectors with the biggest eigenvalues. Then the $(m+1)$-th weight vector $\mathbf{w}_i$ converges to eigenvector $\mathbf{e}_k = \mathbf{e}_{m+1}$ with $\lambda_k < \lambda_p$, $p = 1\ldots m$ and not to one of the eigenvectors with a bigger eigenvalue λ_p, if and only if with eqn (D.2) the ratio

$$\frac{|a_{ik}(t+1)|}{a_{ip}(t+1)|} = \frac{|a_{ik}(t)|\,|1+\gamma\lambda_k\,(1-\beta\sum_{j\neq i} a_{jk}(t)b_{ij}\,/\,a_{ik}(t))|}{|a_{ip}(t)|\,|1+\gamma\lambda_p\,(1-\beta\sum_{j\neq i} a_{jp}(t)b_{ij}\,/\,a_{ip}(t))|} > \frac{|a_{ik}(t)|}{|a_{ip}(t)|}$$

holds in a certain small environment of $\mathbf{e}_k$. In this case, the component a_{jk} of an already converged weight vector $\mathbf{w}_j$ relative to the $(m+1)$-th (orthogonal!) eigenvector $\mathbf{e}_k = \mathbf{e}_{m+1}$ is zero. This means that

$$a_{jk} = 0, \qquad \forall j\in[1\ldots k-1] \qquad \Rightarrow \sum_{j\neq i} a_{jk}b_{ij}/a_{ik} = 0$$

$$a_{j|} = \begin{cases} 0 & j \neq \ell \\ 1 & j = \ell \end{cases} \qquad j,|\in[1\ldots k-1] \qquad \Rightarrow b_{ij} = a_{ij}\lambda_j$$

which is also true for a_{jp}, i.e., when $|$ is denoted as p.

Therefore, we have

$$\sum_{j\neq i} a_{jp}b_{ij}/a_{ip} = \sum_{j\neq i} a_{jp}a_{ij}\lambda_j/a_{ip} = a_{pp}a_{ip}\lambda_p/a_{ip} = \lambda_p$$

and the necessary convergence relation becomes finally

$$\frac{|1+\gamma\lambda_k|}{|1+\gamma\lambda_p\,(1-\beta\lambda_p)|} > 1 \qquad (2.22)$$

For $1-\beta\lambda_p > 0$ or $\beta < 1/\lambda_p$, this means

$$1+\gamma\lambda_k > 1+\gamma\lambda_p(1-\beta\lambda_p) \quad \text{or} \quad \lambda_k > \lambda_p - \beta\lambda_p^2$$

which does not depend on the learning rate but only on the relation

$$\beta > \beta_1 = (\lambda_p-\lambda_k)/\lambda_p^2 \qquad (2.23)$$

and is our well-known relation (2.19). So, if condition (2.22) is at no time satisfied because of a cross-correlation factor $\beta < \beta_1$ which is too small, the convergence of $\mathbf{w}_k$ to a not already existing eigenvector $\mathbf{e}_k$ is impossible. This is especially true for negative β.

For $1-\beta\lambda_p < 0$, i.e., $\beta > 1/\lambda_p$, the value of the whole term can become negative and the sign of the component can alternate after each iteration. Nevertheless, even the oscillating weight vector will converge to eigenvector $\mathbf{e}_k$ if

$$1 + \gamma\lambda_k > -(1 + \gamma\lambda_p(1-\beta\lambda_p)) \quad \text{or} \quad 2 > \gamma(\lambda_p(\beta\lambda_p - 1) - \lambda_k)$$

and thus
$$\gamma < 2/(\beta\lambda_p^2 - (\lambda_p+\lambda_k)) \quad \text{for} \quad (\beta\lambda_p^2 - (\lambda_p+\lambda_k)) > 0 \qquad (2.24a)$$
$$\gamma > 2/(\beta\lambda_p^2 - (\lambda_p+\lambda_k)) \quad \text{for} \quad (\beta\lambda_p^2 - (\lambda_p+\lambda_k)) < 0 \qquad (2.24b)$$

Since (2.24b) always holds for $\gamma > 0$, with eqn (2.24a) there is only one limit β_2 with $(\beta_2\lambda_p^2 - (\lambda_p+\lambda_k)) = 0$ or

$$\beta_2 = (\lambda_p+\lambda_k)/\lambda_p^2 \qquad (2.25)$$

If $\beta > \beta_2$ the condition (2.24a) for γ must be satisfied, otherwise the convergence will not achieve different eigenvectors.

Now, we can summarize the necessary parameter values for convergence to different eigenvectors:

$0 \leq \beta \leq \beta_1 = (\lambda_p-\lambda_k)/\lambda_p^2$ *no convergence to different eigenvectors possible*

$\beta_1 < \beta \leq \beta_2 = (\lambda_p+\lambda_k)/\lambda_p^2$ *convergence with no constraint on* γ

$\beta_2 < \beta$ *convergence* if $\gamma < 2/(\beta\lambda_p^2 - (\lambda_p+\lambda_k))$

In a small environment around $\mathbf{e}_k$, the values are also sufficient.

Is there a universal set of parameters β and γ for all eigenvalues which guarantee the convergence to the different desired eigenvectors for a PCA? With the previous results, we can not rely on the parameter regime $\beta_2 > \beta > \beta_1$ because for certain eigenvalues $\lambda_k \ll \lambda_p$ the parameter β_2 can become maximally $\beta_2 \approx 1/\lambda_{max}$ which is not always bigger than $1/(4\lambda_{min})$ as required by eqn(2.20).

Thus, we have to consider the other possible interval $\beta > \beta_2$ with the weight vector component a_{ik} alternating in sign. For $\lambda_k \approx \lambda_p$, this transforms to $\beta > 2/\lambda_p$. Thus, to guarantee the different fixpoints for all pairs of eigenvalues λ_p, λ_k and all values of λ_p we have to choose the parameters to satisfy the relations

$$\beta > 2/\lambda_{min} \quad \text{and by (2.24a)} \quad \gamma < 2/\lambda^2_{max} \qquad (2.26)$$

as necessary conditions for a proper convergence to all eigenvectors with different eigenvalues.

3. CONVERGENCE SIMULATION

For demonstration purposes let us regard the simulation of a picture processing procedure. Since our con-

verged model with $u_{ij} = 0$ contains only non-coupled, linear neurons in the subpicture processing network, this part of the system implements a simple linear transformation.

Let us concentrate on the more interesting part of the system: the learning of the Karhunen-Loéve transformation. Since the highlight of this transformation is the adaptation to the sensor signal statistics, for picture processing we have to choose a representative picture statistic.

3.1 The training data

There have been many attempts to model the statistics of natural pictures, see e.g. Habibi and Wintz (1971). One of the most useful is by the autocorrelation function between the pixels $\mathbf{x}^1$ and $\mathbf{x}^2$

$$C(\mathbf{x}^1,\mathbf{x}^2) = \exp(-a|x_1^2 - x_1^1| - b|x_2^2 - x_2^1|) \quad a \approx 0.2,\ b \approx 0.1 \qquad (3.1)$$

The form of the analytical solutions for this case are known (Habibi & Wintz, 1971); the two-dim. eigenfunctions are products of cosines and sinus functions determined by the eigenvalues. In the discrete case, the set of samples of these eigenfunctions are the eigenvectors of the autocorrelation matrix and have to be numerically constructed to serve as a reference for the convergence error.

For the picture material presented in Habibi and Wintz (1971) the two coefficients, a for horizontal correlations and b for vertical correlations (measured in 1/pixel-length-units), are different, reflecting the flat, ordered arrangements of artificial building or pictures containing an horizon. Such a horizontal-vertical orientation does not always exist in natural pictures (e.g. trees) which do not contain horizontal or vertical lines. So, let us conveniently assume that we have $a = b$ which covers the correlations in all directions uniformly by the Euclidean distance $|\mathbf{x}^1-\mathbf{x}^2|$ of the two pixels

$$C(\mathbf{x}^1,\mathbf{x}^2) = \exp(-a|\mathbf{x}^1-\mathbf{x}^2|) \qquad (3.2)$$

How can the autocorrelation matrix be constructed ? The autocorrelation matrix of the two-dim. picture matrix will be a four-dim. tensor. To remain in our ordinary notation and to use our ordinary numerical procedure for calculating the eigenvectors, we will instead construct an ordinary two-dim. autocorrelation matrix. For this purpose we concatenate all the n rows of n pixels x_{hk} to one vector $\mathbf{c}$

$$\mathbf{c} = (x_{11}, \ldots, x_{1n}, x_{21}, \ldots, x_{2n}, \ldots, \ldots, x_{n1}, \ldots, x_{nn})$$

The expected autocorrelation $\langle \mathbf{c}\mathbf{c}^T \rangle$ of this vector forms the autocorrelation matrix $\mathbf{C} = (C_{ij})$ where every entry is of the form

$$C_{ij} = \langle x_{kh}x_{st} \rangle = \exp\big(-a|(k,h) - (s,t)|\big) = \exp\big(-a\,((k-s)^2+(h-t)^2)^{1/2}\big)$$

Thus, to construct a $n^2 \times n^2$ correlation matrix, the Euclidean distances between all the pixels in the prior picture frame must be computed and filled in the matrix. This contains ones in the main diagonal and is symmetric; the smooth value changes outside the diagonal are regularly broken.

3.2 A Convergence Example

For the case of $m = 4$ neurons the non-linear convergence in the learning of the network is demonstrated. For 3×3 pictures, we have a 9×9 autocorrelation matrix with 9 eigenvectors and 9 eigenvalues. For $a = 0.2$, the maximal eigenvalue is $\lambda_{max} = \lambda_1 = 6.81414$, $\lambda_2 = \lambda_3 = 0.639568$ and $\lambda_4 = 0.22733$. According to eqn (2.26) we choose $\beta = 9.1 > 2/\lambda_4 = 8.8$ and $\gamma = 0.043 < 2/\lambda_1^2$. The weight vectors are randomly initialized and normalized to length 1. For the deterministic iteration by eqn (2.12) we use the correlation matrix $\mathbf{C}_{xx}$ obtained in the previous section.

Figure 6 shows the convergence of the four weight vectors by their normalized projections $\cos(\mathbf{w}_i,\mathbf{e}_k) = \mathbf{w}_i^T\mathbf{e}_k/|\mathbf{w}_i||\mathbf{e}_k|$ on the corresponding eigenvector, i.e., the cosines between the weight vector and the eigenvector, for $t = 1,\ldots,5000$ iterations on a logarithmic scale.

Since the second and third eigenvalues are equal, all possible linear combinations of the corresponding eigenvectors are also eigenvectors. Instead of one direction, every vector of the whole plane $\mathbf{p}_{23} = a_2\mathbf{e}_2 + a_3\mathbf{e}_3$ spanned by the two eigenvectors $\mathbf{e}_2$ and $\mathbf{e}_3$ is an eigenvector and therefore a convergence goal.

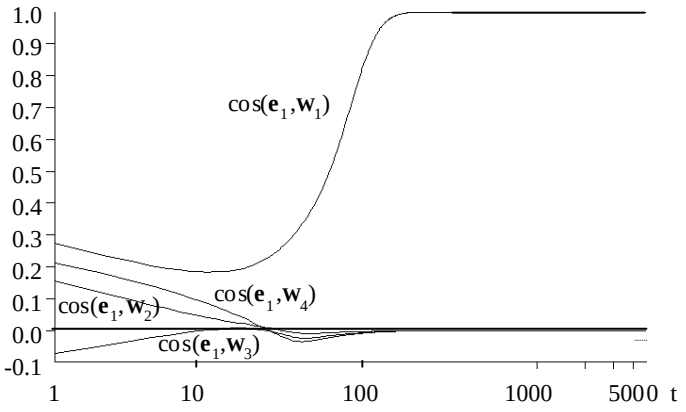

Fig. 6 *The time course of the weight vector projections on eigenvector e1.The convergence to the eigenvector with the biggest eigenvalue is marked by a relative fast speed due to the strong autocorrelation feedback in the learning process.*

Thus, the convergence can be measured by the projection length $\cos(\mathbf{w}_i,\mathbf{p}_{23})$ of a weight vector into this plane which is in this case $\cos(\mathbf{w}_i,\mathbf{p}_{23}) = |(\mathbf{w}_i^T\mathbf{e}_2)\mathbf{e}_2 + (\mathbf{w}_i^T\mathbf{e}_3)\mathbf{e}_3| / |\mathbf{w}_i|$.

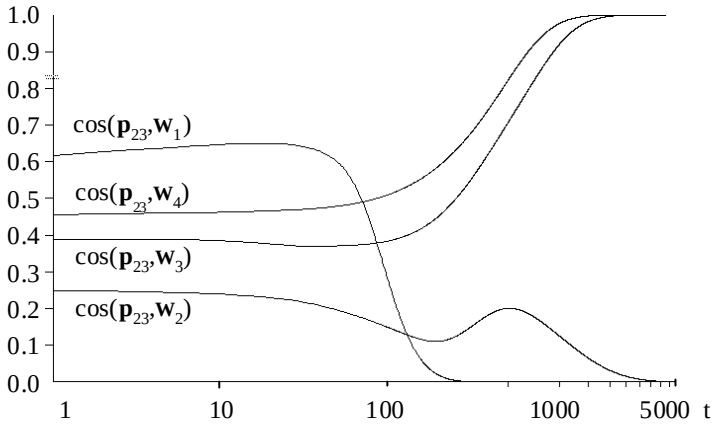

Fig. 7 *The time course of the weight vector projections on eigenvector plane* $\mathbf{p}_{23}$*. Since for the specific picture statistics the second and third eigenvalues are the same, each linear combination of two eigenvectors is also an eigenvector. The convergence can only be measured by the projection on the plane spanned by the two eigenvectors and is approximately ten times slower than for the biggest component which scales well with the ratio of the eigenvalues*

We see that the convergence is not straight and simple; there are quite complicated "movements" of the weight vectors in the input space. Comparing Fig. 6 with Fig. 7 we can observe that the convergence to the eigenvector with the biggest eigenvalue is the most rapid one. This can be explained by the strong variance (principal component) in the eigenvector direction which results in strong Hebbian terms of eqn (2.11) and therefore in strong changes in that direction.

Fig. 8 shows the convergence of the weight vector to the eigenvector with the smallest eigenvalue in the set of the four biggest eigenvalues.

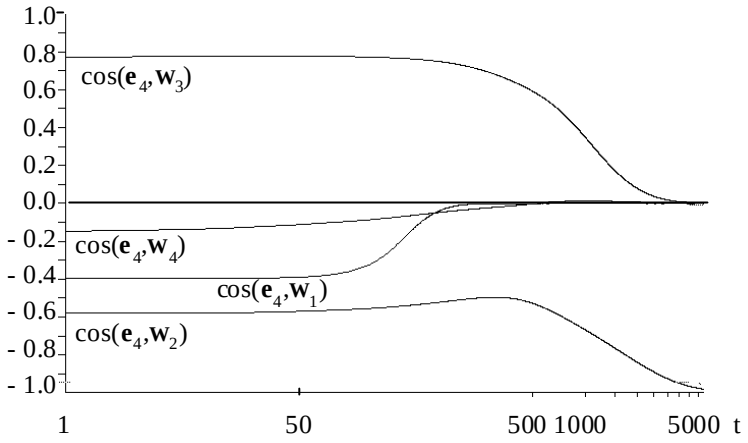

Fig. 8 *The time course of the weight vector projections on eigenvector e_4. The convergence is not as fast as in Fig. 6 or Fig. 7, showing the small influence of the eigenvalue.*

We see that the more the eigenvalues are equal, the convergence speed rapidly decreases.Now, are there means to speed up the convergence process ?

3.3 Growing lateral inhibition

We know that we can not change the parameter regime very much yet still insuring the convergence of the system. Nevertheless, for the biological counterpart we know that the main structure of the neurons are genetically preset and develops during the maturing of the nervous system (Kuffler, Nicholls, Martin 1984). This is only true for the raw structure. The important fact in the neural developement is the additional growth of the neural synapses and dendrites due to some data-specific, build-in pattern processing algorithm which we do not yet know. Nevertheless, we do know that in these systems the lateral network connections, and therefore also the inhibitions, grow with time to an important amount. What does this mean for our lateral inhibition network?

We know for uncoupled neurons ($\beta = 0$), that each neuronal weight vector will converge independently to the eigenvector with the biggest eigenvalue λ_{max}. Only by the cross-correlation influence of the lateral inhibition are the weight vectors driven to different eigenvectors. If we start with a small lateral inhibition β the system should be oriented towards the eigenvector with the biggest eigenvalue. On the basis of this, augmenting β should cause a readjustment of the system on basis of an already found eigenvector and should speed up the convergence for the rest of the weight vectors.

This scheme for β is shown in Fig. 9.

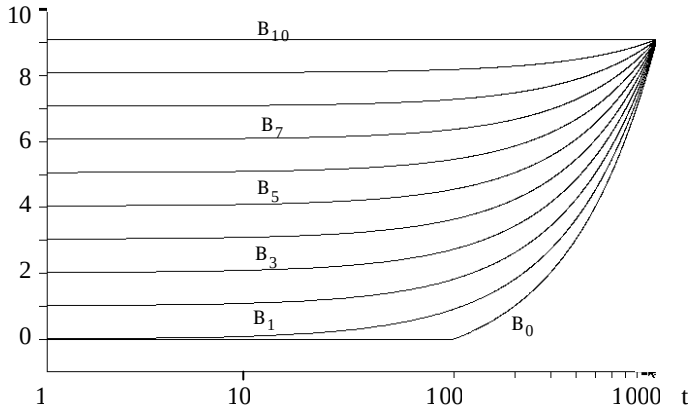

Fig. 9 *The increase functions of the lateral inhibition factor b. The linear functions are drawn on a logarithmic scale to reflect the scaling of the convergence plots of Fig. 10.*

Here we have 11 different linear functions for $\beta(t)$, denoted by $\beta_i(t)$ and shown on a logarithmic scale. All of them reach the necessary value $\beta = 9.1$ (see above) for t = 1000. This 11 different functions are used to iterate the same system of m = 4 neurons of the previous section. The result is shown in Fig. 10 where the objective function $R(\beta_i,t)$ is plotted for five different functions $\beta_i(t)$, $i = 0,1,2,5,10$.

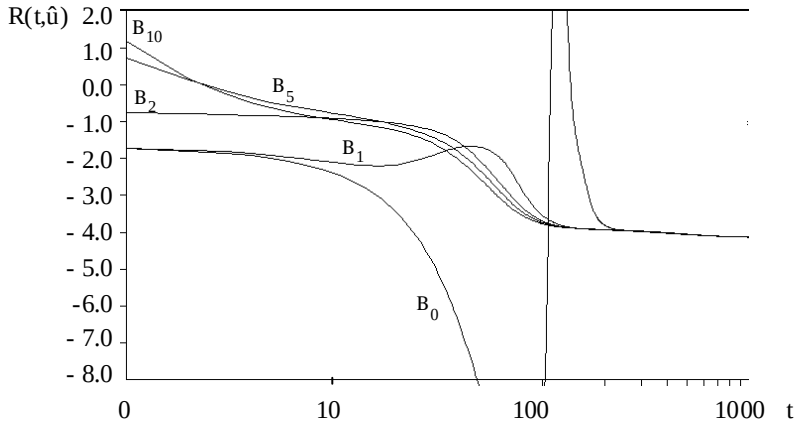

Fig. 10 *The objective function time course plotted for different increasing lateral inhibition functions. You see that small inhibital connections are necessary to avoid unnecessary oscillations (b_0,b_1) of the learning process. Nevertheless, the convergence itself is fast stabilized and remains stable throughout a broad regime of lateral inhibition growth conditions.*

As we can see, the increase of the lateral inhibition supports the convergence for a certain degree and yields better results as the constant inhibition $\beta_{10}(t) = \text{const} = \beta$ for the beginning of the learning process. Nevertheless, if we prohibit the inhibition too long (e.g. like β_0), the network converges too much in the wrong direction and the new orientation slows the convergence down. For optimal results, the function $\beta(t)$ should increase more than linear. An exponential increase in $\beta(t)$ which models the biological growing of the axons and dendrites should provide better results.

Up to now, the optimal inhibition function $\beta(t)$ for the model which depends on the parameter set of the input statistics is still unknown and subject to future research.

4. SELF-ORGANIZATION OF A CELLULAR NEURAL NETWORK

The lateral inhibited network, introduced in section 2, is often used in biologically motivated models of nerve functions. One of the most popular ones is the self-organizing map of Kohonen (1982) which is based on the model of Willshaw and von der Malsburg (1976) and has been analyzed by Amari (1980). In this section we show how the previous introduced symmetric model can be used to implement a new kind of self-organization which can be seen as a generalization of the Kohonen map.

After we have shown the relations of this popular self-organizing network to the one we have developed so far, we will show that our model also leads to a kind of self-organization, related to but essentially different to the former one. Additionally, we will show how this self-organization leads to a two-dim. encoding of pictures which can be used in a VLSI implementation of the transform coding model by a special kind of net, the so-called “cellular neural networks".

Now, let us first regard the relations of the self-organizing Kohonen map to the previously introduced lateral inhibition model.

4.1 The Shortcut Algorithm of the Kohonen Map and the Eigenvector Jets

Let us assume that we have one sensor layer providing the input and another layer of processing neurons. The processing neurons have locally distributed, lateral connections u_{ij} and are located at a position $\mathbf{v}_i$ in the output space. The situation is shown in figure 4.1.

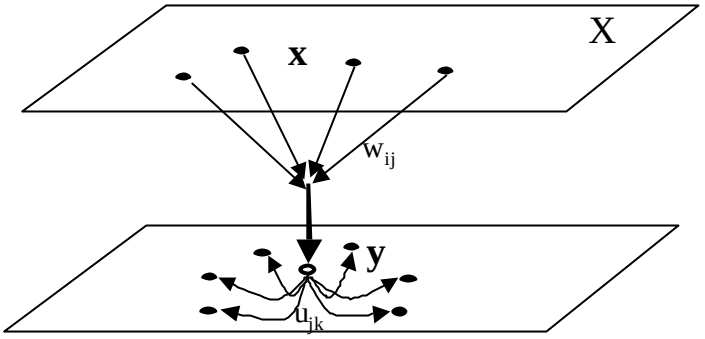

Fig. 11 *A locally lateral connected continuous neural layer field. Every neuron in the continuous neuron field (lower layer) receives its input from a local region of a continuous input field X (upper layer) and influences its neighbors locally by its output y.*

The lateral inhibited, self-organizing network activity is in a discrete form (see e.g. Kohonen 1988b) for neuron *i*

$y_i = S(z_i)$ $S(.)$ = *activation function, squashing function*

with $z_i = \sum_j w_{ij}x_j + \sum_j u_{ij}y_j = \mathbf{w}_i^T\mathbf{x} + \mathbf{u}_i^T\mathbf{y}$ (4.1)

modelling the threshold by a special weight and a constant input line. For linear neurons with $y_i = S(z_i) = z_i/2$ and normalized self-exitation $u_{ii} = 1$ eqn (4.1) becomes eqn (2.1). Thus, our activity model of section 2 is a special case of the of the self-organizing neural layer activity model. The main difference lies in the generally non-linear squashing function S(.). With fixed lateral inhibition weights according to a ON centre/OFF surround (the mexican hat function) this leads to a grouped winner-take-all mechanism (*population encoding, "bubbles"*) for the self-organizing neural layer at normalized input (Kohonen 1993), whereas our linear model just gives a linear response.

The learning eqn for neuron *i* in the self-organizing neural layer is a Hebbian term, reduced to adaptively normalize the weights by a "forgetting term" $f(.)\mathbf{w}_i$, see (Kohonen,1989) .

$$\mathbf{w}_i(t+1) = \mathbf{w}_i(t) + \gamma\,(y_i\mathbf{x} - f(y_i)\mathbf{w}_i(t)) \qquad (4.2)$$

Conveniently, the continuous function f(.) has only to meet the conditions f(0) = 0, f(1) = 1. In the middle of the activity group, the output becomes saturated with $y_i = S(z_i) = 1$, $f(y_i) = 1$ and the activity (4.1) becomes with constant $\mathbf{u}_i^T\mathbf{y}$ at the centre neuron *c*

$$z_c = \mathbf{w}_c^T\mathbf{x} \text{ with } \mathbf{w}_c^T\mathbf{x} = \max_i \mathbf{w}_i^T\mathbf{x} \qquad \textit{correlation winner-take all rule} \qquad (4.3)$$

For a given input $\mathbf{x}$ and normalized weight vector length $|\mathbf{w}_i|$, this corresponds by the equation $(\mathbf{x}-\mathbf{w}_i)^2 = |\mathbf{x}|^2 - 2\mathbf{w}_i^T\mathbf{x} + |\mathbf{w}_i|^2$ to the selection rule

$$|\mathbf{x}-\mathbf{w}_c| = \min_i |\mathbf{x}-\mathbf{w}_i| \qquad \textit{distance winner-take all rule} \qquad (4.4)$$

and the learning rule for the active neurons, including the centre neuron becomes

$$\mathbf{w}_i(t+1) = \mathbf{w}_i(t) + \gamma\, h(i,c,t)(\mathbf{x} - \mathbf{w}_i(t)) \qquad (4.5)$$

where the neighbourhood function h(i,c,t) implements a kind of mexican hat function by the fixed lateral weights u_{ij}. For example, the neighbourhood function could be constant with value one for all neighbours around neuron c, otherwise zero. Or, we can take it as the Gaussian of the neural output position $\mathbf{v}$ as $h(i,c,t) = \exp\left(-(\mathbf{v}_c-\mathbf{v}_i)^2/\sigma^2(t)\right)$.

By this, the Kohonen map describes the learning of the centre neuron *c*. Eqns (4.2) and (4.5) do not describe the same situation and cannot be directly interchanged (see Acker and Kurz, 1990). Eqn (4.2) describes the change of weights at implicitly normalized weights, whereas eqn (4.5) assumes this situation and describes the change of weights without referring to normalization any more. Thus, the self-organized map induced by (4.5) forms itself on the hypersphere with radius 1 of the weight vectors of eqn (4.2), whereas the weight vector of eqn (4.5) has one dimension less: it lacks the offset parameter of the threshold, giving not normalized weight vectors, and describes the coordinates just on the hypersphere.

For the Kohonen map, the weight vector can be seen as a the *class prototype* of a *classification* process which results in clustering the input space. The iterative algorithm of (4.5) which narrows down the neighborhood in time to one neuron lets the weight vector converge to the least mean squared error solution for $R(\mathbf{w}) = \langle(\mathbf{x}-\mathbf{w}_i)^2\rangle$ of the set of local input patterns. Thus, similar to section 1.2, in the limit case for normalized weight vectors each class prototype represents the eigenvector of the local cluster on the hypersphere with the biggest eigenvalue. The whole self-organization process arranges a kind of non-linear PCA in the input space: the tessellation is a patchwork of local, linear PCAs which represent at each place the cluster mean vector as the most important feature.

Now, the remaining sections try to show that this can be extended to yield in each region not only the principal eigenvector, but also the whole set of the eigenvector decomposition coefficients. Here, as patches of the input space we consider the non-overlapping subspaces of the original input space, i.e., the subpictures of Fig. 3.

First, we introduce a discrete version of a continuous neuronal field which has become very popular by multiple application and implementation efforts: the cellular neuronal network. Then, using this model, we will show how we can derive locally self-organized *eigenvector jets*, representing a new kind of self-organization.

4.2 Cellular Neural Nets

A kind of modular organization for the activity phase of lateral inhibited networks has been coined by Chua and Yang (1988) with the term *cellular neural networks* and has been adopted by an international group of scientists as a paradigm for a supercomputer mainly used for image processing (Roska 1993, TCS 1993). Here, the weights of the neurons (*templates*) are set arbitrarily by the user and can be seen as a form of programming.

One of the main new ideas of this paradigm of neural networks is the devotion of a neuron to only "local" data processing, restricted to a subset of all possible input lines. This idea fits well to the needs of VLSI design which favors building big systems by the replication of small, modular, local functions. Since the VLSI design is normally implemented on a two-dim. wafer, the approach is well suited for two-dim. sensor fields, e.g. for image processing. Nevertheless, the networks can also principally used in a one-dim. or three-dim. design or any other number of neighborhood dimensions. A typical input layout is shown in Fig. 12. Here, only the sensor elements (disks) and the neurons (rectangles), but no output lines are shown.

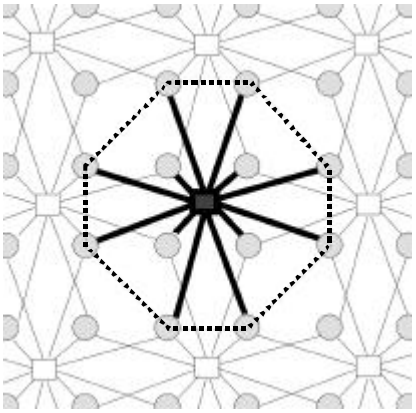

Fig. 12 *An example of a two-dim. cellular neural network. Here, shaded disks denote the sensor elements (like light sensitive transistors), rectangles denote the artificial neurons and the lines are the input connections to them. For clarity, the output connections are not shown. The thickened lines are the input connections for one neuron; the dotted line denotes its receptive input field.*

More formally, a CNN can be defined as a *n*-dimensional array of identical dynamical systems (cells), which have most interactions locally within a finite radius *r* and have only continuous valued state variables, see Chua and Roska (1993). The input connections of a cell with index *(i,j)* can be described by a matrix $\mathbf{B}^{ij}$, the local interconnections between the other cells by a matrix $\mathbf{A}^{ij}$. Both matrices are referenced as "templates". In general, the activity influence is a non-linear function of these weight matrices. All connections are real-valued and can also model time delays. The internal activity state is modelled by an "evolution law" (ODE, differential/difference equations, functional maps, etc.) and includes the input and interconnection influences.

As an example, let us consider a simple, linear feedforward network where each neuron computes the linear sum of the input values of 9 image pixels. The differential equation for the state of such a neuron is given, using the original notation by the expression

$$\partial z_{ij}(t)/\partial t = - z_{ij}(t) + \sum_{k,l \in N(i,j)} A^{ij}_{kl}\, y_{kl} + \sum_{k,l \in N(i,j)} B^{ij}_{kl}\, x_{kl} + I_{ij}$$

N(i,j) = neighbourhood of cell (i,j)

and by the matrices

$$\mathbf{A}^{ij} = \begin{bmatrix} u_{11} & u_{12} & u_{13} \\ u_{21} & u_{22} & u_{23} \\ u_{31} & u_{32} & u_{33} \end{bmatrix} \qquad \mathbf{B}^{ij} = \begin{bmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \\ w_{31} & w_{32} & w_{33} \end{bmatrix} \qquad I_{ij} = 0$$

where the matrix coefficients u_{kl} and w_{kl} generally depend on the location, i.e., on the cell indices *(i,j)* with $u_{kl} = u_{kl}(i,j)$ and $w_{kl} = w_{kl}(i,j)$.

Since the matrix $\mathbf{B}^{ij}$ is used at all sensor points, it can be seen as a local picture processing operator which is identical to the operators used in conventional image processing, see for example Ballard and Brown (1982). Thus, a chip containing an array of CNNs performs like a high-speed image processing supercomputer, having a performance of 10^{12} = 1000 GOPS (Giga operations per second) in currently available technology (Roska, 1993).

If the connection templates are identical, they are called "cloning templates". Although the templates are often cloned and fixed, this is not necessarily the case. In the next section, the use of the neural network of section 2 shows this for the self-organized developement of non-identical input templates.

4.3 Simulation of the Self-organization Process

Let us consider a symmetrical, lateral inhibited network as it has been introduced in section 2. Additionally, let us have only a discrete, limited radius *R* of inhibition influence as it is defined for CNNs. For a one-dim.

neighbourhood network, as used for instance for the coding of time signals x(t) discretized by a tapped delay line, the interconnection scheme for $R = 1$ and $R = 2$ is shown in Fig. 13.

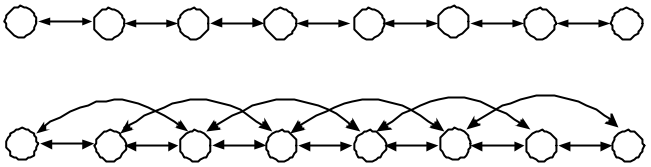

Fig. 13 *One-dim. lateral inhibition interconnections. The double-arrowed lines show the lateral inhibition influence between a neuron and its neighbors. Neither output nor input lines are shown.*

The simulation used input patterns of $n = 36$ components, each one set as a random variable by independent Gaussian noise with different variances. The input weights for $m = 8$ neurons are randomly initialized with a fixed vector length $|\mathbf{w}_i| = 1$, the lateral weights are initialized with zero. The parameters β and $\gamma(t = 0)$ are set according to eqn (2.26) with decreasing $\gamma(t)$.

The result of a simulation is shown in Table 1. Here, the index of the approximated eigenvector, denoted by the order of the corresponding eigenvalues, is listed for an inhibition radius of $R = 1$.

Table 1 The Goal of Convergence for $R = 1$

Neuron	1	2	3	4	5	6	7	8
Eigenvector index	1	2	1	2	1	2	1	2

How can this result be explained? The eigenvector with index 1 is the one with the biggest eigenvalue $\lambda_1 > \lambda_2$. Therefore, each neuron tries to converge to eigenvector 1 and will do this if no lateral connections exist. If two neurons have mutual lateral inhibition, the one having the initial weight vector most similar to eigenvector 1 will win the competition and converge to it, disabling all other neurons connected to it to converge to this eigenvector. Thus, the radius $R = 1$ disables the neighbours to converge to eigenvector 1, leaving them only the possibility to converge to the one with the next smaller eigenvalue which results in an alternating order of eigenvectors. In Table 2 the results of similar simulations are shown, but with $R = 2$. It shows two simulation runs, each one starting with randomly initialized weights.

Table 2 The goal of convergence for $R = 2$

	Neuron	1	2	3	4	5	6	7	8
run 1:	Eigenvector	1	2	3	1	2	3	1	2
run 2:	Eigenvector	2	1	3	2	1	3	2	1

Here, the same considerations as for Table 1 are valid. Within the enlarged radius we know that all other eigenvectors can exist except the one to where the centre neuron weight vector will converge. This is valid for all neurons. Thus, different sets of eigenvectors can be observed; the PCA is performed by local groups of neurons. For example, in Table 2 for run 1 the sets are {1,2,3}, {2,3,4}, {3,4,5}, {4,5,6}, {5,6,7}, {6,7,8}. The coefficients of this local base vectors decomposition can be termed *eigenvector jets* analogously to the well known "Gabor jets" of local Fourier transform (Buhmann et al., 1989).

Now, let us extend this model to the important case of two dimensions, for instance for image encoding on the sensor chip by cellular neural nets. In Fig. 14 and Fig. 15 two networks of $m = 16$ neurons enlarged by additional two-dim. horizontal and vertical connections are studied, one with $R = 1$ and one with $R = 2$. The simulation results are also shown in these figures.

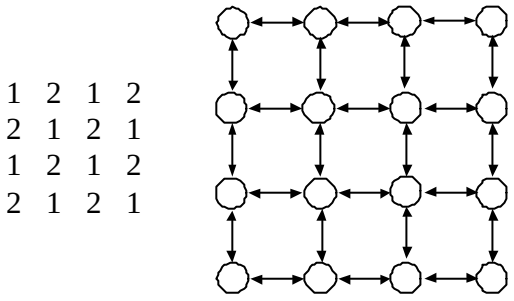

Fig. 14 *The mapping of eigenvector indices to neural locations after simulated inhibition and the net structure of the lateral inhibitions for the lateral inhibition distance R = 1. The index matrix shows only two alternating indices. This structure is shifted for different simulation runs, but remains principally the same.*

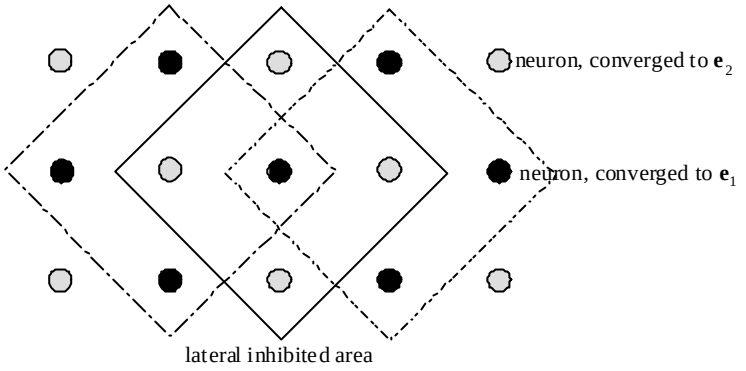

Fig. 15 *The sites of possible neighboured neurons of the same component. The alternating structure of the index matrix in Fig. 14 can be explained by inspecting the possible eigenvector indices within the inhibition radius of one neuron, showed as a squared line. If we assume the convergence to the eigenvector with the next eigenvalue index in horizontal directions, the same goes for those neurons within its inhibition fields. Since this argument is also valid for the vertical direction, the alternating structure in both dimensions is the one which is stable and yield the eigenvectors with the biggest eigenvalues, i.e., a minimum of the global risk function.*

It is evident that the eigenvector index mappings are direct extensions of Table 1 and 2. It is also clear that other shifts in one row or column of the basic pattern can be observed.

In Fig. 16 the situation of Fig. 14 with the inhibition neighborhood of one neural unit is shown for a geometrical discussion of the organization process.

1 2 3 1	2 3 1 2	2 1 3 2
3 1 2 3	3 1 2 3	1 3 2 1
2 3 1 2	1 2 3 1	3 2 1 3
1 2 3 1	2 3 1 2	2 1 3 2

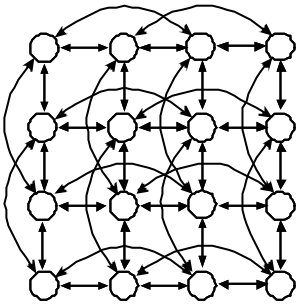

Fig. 16 *The formation of local eigenvector sets and the lateral inhibitions for the horizontal and vertical inhibitions with R = 2. In contrast to Fig. 15, the vertically and horizontally increased inhibition suppresses the same eigenvector indices within the inhibition radius and cause the appearance of other eigenvectors with smaller eigenvalues. The index assignments are listed for different simulation runs .*

The convergence result is indicated by the full black unit (eigenvector 1) or the shaded unit (eigenvector 2). As already argued before, the inhibition enables the convergence of a weight vector with the same index only outside the reach of the inhibition. For $R = 1$, this leads to an alternating formation of eigenvectors in both horizontal and vertical directions and generates a chessboard-like appearance. Here again, we see that the formation of local eigenvector sets is automatically obtained by the existence of the discrete inhibition radius.

In figures 17 and 18 diagonal inhibition connections are added to the nets and we can observe the mapping of eigenvector indices to neural location convergence configurations after different runs with randomly initialized weights.

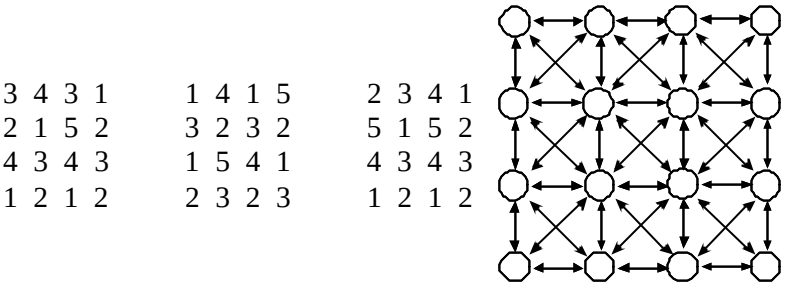

Fig. 17 *The formation of local eigenvector sets and the lateral inhibitions for the horizontal, vertical and diagonal inhibitions with R = 1.The additional diagonal inhibition also enlarges the inhibition radius and has similar effects as in Fig. 16.*

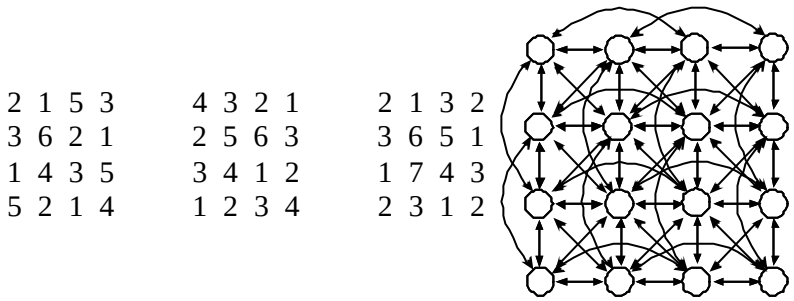

Fig. 18 *The formation of local eigenvector sets and the lateral inhibitions for the horizontal, vertical and diagonal inhibitions with R = 2.*

Here again, an enlarged inhibition radius forces the convergence of the other neurons within the radius to eigenvectors with smaller eigenvalues, enabling the self-organized formation of two-dimensional eigenvector jets. Simulations with other radii confirm the principal mechanisms.

Although the whole input is received by all neuronal units as it is the case in the Kohonen map, the same results can be obtained for discrete CNN systems with restricted localized input regions (local receptive fields) if the input statistics are translation-invariant. For most data like speech and image this is the case, because the neighbour data points are more correlated than ones with a longer distance, independent of the absolute position in time or picture coordinates. Although the localized input leads to localized statistics which produces no more classical Karhunen-Loéwe transforms because the input set is different for all components, we can obtain the same classical transformation results. Experiments for this postulation are under development.

There is another important remark. Due to the randomized initial conditions, the weights of the neurons can converge in all simulations in a transparent manner. Nevertheless, let us assume for instance as special initial weight conditions that the weights of neurons 1 and 4 in Fig. 13 have both converged to eigenvector $\mathbf{e}_1$. Then, for $R = 1$ under the assumption of only small random perturbations caused by the input variance, there is no reason why the weights of neurons 1 and 4 should change their fixed points: all perturbations by the lateral inhibitions are smaller than the autocorrelation part, leaving the weights at the "strongest" fixed points. Thus, the weights of neurons 2 and 3 will only evolve to the less stronger eigenvectors $\mathbf{e}_2$ and $\mathbf{e}_3$, and not to the configuration of Table 1 with the lowest possible objective function value or "energy".

This irregularity can be compared with the formation of regular crystal lattice by atom bindings: due to thermal effects (random perturbations) there is no global coordination of the local ordering process which results in the effect of local crystal disorders. For large, randomly initialized arrays of neurons (which we cannot yet simulate), this effect should lead to analogue observations.

Now, let us take a closer look at the question: what are the "native", fault-free structures of this kind of self-organization with lowest energy? If we can already deduce the two-dim. structure of the stable configurations, we can artificially initialize it as ROM on the chip in order to provide complete local sets of eigenvector decomposition coefficients for the best distributed picture encoding and decoding possible by this kind of architecture.

4.4 The Proportions of the Self-organization

For the self-organizing process as a whole, there are many questions still unresolved. For instance, given the inhibition radius, how many stable states are principally possible? Are the different components of the eigenvector jets always equally distributed?

In this section, we want to focus on some of theses questions and try to sketch the design of an optimal eigenvector jet system. Since the regions of influence of the lateral inhibitions are sharply limited, the question of

global convergence reduces to the question if a specific assignment of eigenvectors to weight vectors are consistent or not and can be solved by the following geometrical reasoning.

Let us consider an array of neurons, very densely packed, i.e., a nearly continuous field. In this field, every neuron inhibits the formation of the same eigenvector within the field bordered by a circle of radius *R*. Thus, if we have n_F eigenvector jets per area unit, a first rough estimation yields at most $N_{jet} = n_F A = n_F \pi R^2 \sim R^2$ other eigenvector components within this radius.

Now, let us evaluate this more accurately. Let us regard one neuron with its lateral inhibition of radius *R* (see Fig. 19) and let us assume that this neuron has weights which have converged to the first eigenvector $\mathbf{e}_1$, i.e., to the eigenvector with the biggest eigenvalue λ_{max}.

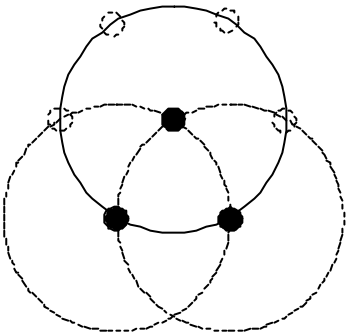

Fig. 19 *The forming of two-dim. discrete structures by lateral inhibition. In a continuous neural field and a discrete inhibition radius, the next neuron with the same eigenvector will lie on the inhibition radius. Selecting arbitrary one point on the radius, a third one should lie outside the first inhibition radius and the second one (dotted line). If we choose the cross points of these two radii as locations for other neural units of the same eigenvector as weight vector, we get first an equal-sided triangle and then end up with a hexagonal structure (solid and dotted line units). Since this is the maximal number of the same kind of units on the inhibition boarder, this structure yields the lowest risk function value of the region; it contains most of the units with the same (maximal) eigenvalue in the nearest neighbourhood of our first unit.*

Then, within its radius *R* no other neuron can converge to $\mathbf{e}_1$. Since it is the dominant eigenvalue, there is at least one neuron beyond the circle of radius *R* which will converge also to $\mathbf{e}_1$. Let us assume that the circle line indicates the site of the first neurons with distance greater than *R* to the centre neuron, then a second neuron is

situated on this circle. Here, again, we have a lateral inhibition area, limited by a circle with radius *R*, shown as a dotted line on Fig. 19. Certainly, on the crossing of the circles of the first neuron and the second neuron a third neuron will exist which converges to the dominant $\mathbf{e}_1$ and has also a lateral inhibition area, shown by a dotted circle. We see now that in this neuronal field, all neurons with the dominant eigenvector are situated at a distance *R* from each other. The first three neurons form an equal-sided triangle, enclosing an angle of 60° at each corner. Since a stable state of lowest "energy" around the first neuron is only possible when a natural number of neurons forms the neighborhood, we have $360^o/60^o = 6$ neurons as neighbors on the circle. Thus, each neuron is the centre of six neighbors, forming a kind of hexagon structure. For the second eigenvector, this is also true. Since it does not interfere with the first eigenvector, there will be also an hexagonal structure build up by those neurons which are inhibited to converge to the first eigenvector because they are located within the inhibited area. This structure can be seen as a "copy" or "shadow structure" of the first one, shifted within the inhibition area.

Thus, the maximal number of components of the eigenvector jets is the maximal number of hexagon copies which can be arranged within the inhibition area using the spare neurons. Since every copy uses more than one spare neuron in the area, there are certainly less components than units in the area. To compute the exact number, let us assume that the inhibition radius is scaled in inter-unit distance lengths. Within an inhibition distance *R* of a unit we can reach *R*-1 other units (drawn in Fig. 20 with different textures at the line crossings) containing different eigenvectors as weights.

Therefore, as a discrete, regular pattern of neuronal locations we assume a regular, two-dim. lattice structure. With two degrees of freedom, this allows us to shift the hexagonal inhibition structure an integer number of increments in the two main index directions x_1 and x_2 maximally *R*-1 times.

As you can see in Fig. 20, all units with negative values of an index x_1 or x_2 are covered by the circle units when we assume positive shifting only. Because the other neurons on the hexagonal circle around the centre neuron have the same eigenvector, the shifting can be restricted to the positive direction of indices x_1 and x_2. Counting all units within the x_1/x_2 area, we have without the circle boarder units (which are the same as those on the x_1/x_2 axis) just R^2 different units

$$N_{jet} = R^2$$

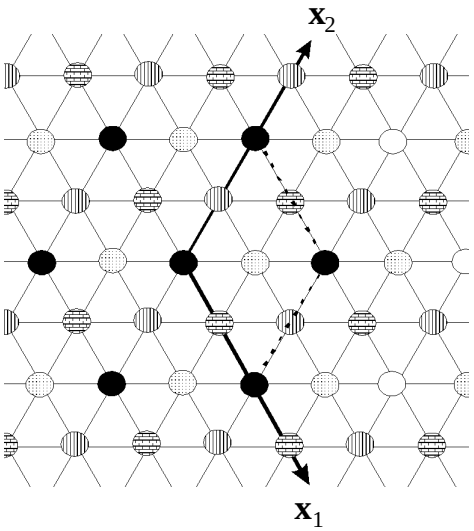

Fig. 20 *The hexagon base structure and the self-organization of eigenvector jets within the structure. The units with the maximal eigenvalue (black and shaded solid disks) form a hexagon structure. Assuming an additional unit between them, the inter-unit inhibition radius is R = 2. This allows a certain number of different eigenvectors for the other units within this radius. To calculate it, we assign two coordinate axes x_1 and x_2 to the two-dim. structure. As we can see, shifting the centre of the hexagonal structure within the region (dotted lines) of {(0,0), (R,0), (0,R), (R,R)} suffices to determine a consistent assignment of eigenvectors indices to all other units within the inhibition radius of the unit (0,0).*

Comparing this with our rough estimation, we get $n_F = 1/\pi$ units per area. At least for this kind of structure, we can predict that regular patterns of convergence can evolve and are stable, providing us with $N_{jet} = R^2$ different eigenvectors on a chip area of R^2. By construction, this is an optimal value. All other, non-hexagonal kind of unit layout pattern structures will yield a smaller number of different eigenvectors, because due to the misfittings of the projection of the dominant hexagon eigenvector structure to the unit layout pattern there will be less unused units for higher components.

The local picture encoding by these eigenvector jets can be compared with the commonly used color dot triples of the matrix in color TV display screen tubes. Those also encode locally the linearly decomposed brightness by three frequency bands, represented naturally by the brightness of the three color dots. Here, the local amplitudes of the three dots can be also seen as a "color jet".

5. CONCLUSION

After an introduction of encoding frequent events and the idea of breaking whole sensor fields like images into small patches, the paper focused on the linear transformation with the smallest achievable error for reproduction of one patch: the transform coding approach using the principal component analysis (PCA).

A new symmetrical, lateral inhibited neural network model for the implementation of principal component analysis is introduced, an objective function for it is proposed from which the corresponding learning rules are deduced. Then the necessary conditions for the learning rate and the inhibition parameter for balancing the cross-correlations and the autocorrelations are computed. The simulation revealed the interesting feature that a slowly increasing inhibition parameter can speed up the convergence process in the beginning.

Finally, the use of non-completely connected, lateral inhibited networks for the self-organized formation of templates in cellular neural networks is shown. Hereby, the classical transform picture coding scheme is changed to a parallel, local model of linear transformation by locally changing sets of eigenvector jets with overlapping input receptive fields which are self-organizing its structure. The well-known Kohonen map can be regarded as the first order version of this more general encoding scheme. Geometrical analysis reveals that the most appropriate structure for this kind of encoding in a plane are arrays of neurons providing hexagonal structures of eigenvector jets.

Our approach shows how an self-organized implementation of sensor encoding can be arranged directly on the sensor or display chip. This enables effective, cheap chip implementations which is an important clue for many applications in real-time image, speech and music encoding for telecommunication, multi-media application and environmental picture data banks.

It should be emphasized that this self-organization does not depend on the specific PCA model introduced in this paper but should be valid also for other models using symmetrical, lateral inhibition connections, e.g., the models of Freisleben (1993), Földiák (1989) or Leen (1991).

Further research will be done on extensions of the schema presented here. For instance, different inhibition radii for different components will naturally lead to a adaptive multi-resolution schema of sensor encoding or an whitening filter network, see Plumbley (1993).

ACKNOWLEDGEMENTS

I am indebted by Jutta Matthes and Andrea Ehlert for the preparation of the simulation results presented in the tables 1 and 2 and figures 14,16,17,18. Additionally, I want to thank one of the referees who by his constructive and helpful remarks markedly improved the correctness and readability of this paper.

6. REFERENCES

R. Acker, A. Kurz (1990). On the biologically motivated derivation of Kohonens Self-organizing feature maps; in: R. Eckmiller, G. Hartmann, g. Hauske (Eds.), **Parallel Processing in Neural Systems and Computers**, Elsevier Sc. Publ., pp. 229-232

S. Amari (1972). Learning Patterns and Pattern Sequences by Self-Organizing Nets of Threshold Elements; *IEEE Trans. on Comp*., Vol C-21, No.11, pp.1197-1206

S. Amari (1980). Topographic organization of nerve fields, *Bulletin of Mathematical Biology*, Vol.42, pp. 339-364

D. Ballard, C.Brown (1982). **Computer Vision**; Prentice-Hall

H. Barrow (1987). Learning receptive fields; *IEEE Proc. Int. Conf. Neural Netw*., pp.IV/115-121

R. Brause (1992a). The Minimum Entropy Neuron- A New Building Block for Clustering Transformations; in: **Art. Neural Networks**, I. Aleksander, J.Taylor (eds.), Elsevier Publ. Comp., pp.1095-1098

R. Brause (1992b). The Minimum Entropy Network; *Proc. IEEE Tools for Art. Intell. TAI-92*

R. Brause (1993a) A Symmetrical Lateral Inhibited Network for PCA and Feature Decorrelation; *Proc. Int. Conf. Art. Neural Networks ICANN-93*, Springer Verlag , pp.486-489

R. Brause (1993b) Transform Coding by Lateral Inhibited Neural Nets; *Proc. 5th IEEE Int. Conf. Tools with AI, TAI-93*

R. Brause (1994). A VLSI-Design of the Minimum Entropy Neuron; in: J. Delgado-Fria, W. Moore(eds.): **VLSI for Artificial Intelligence and Neural Networks**, Plenum Publ. Corp., 1994

I. Bronstein, K. Semendjajew: Taschenbuch der Mathematik; Teubner Verlag, Leipzig 1990

J. Buhmann, J. Lange, C. von der Malsburg: Distortion Invariant Object Recognition by Matching Hierarchically Labeled Graphs; IEEE Int. Conf. on Neural Networks IJCNN, Washington 1989, pp. II/411-416

Y. Chauvin (1989). Principal Component Analysis by Gradient Descent on a Constrained Linear Hebbian Cell; *IEEE Proc. Int. Conf. Neural Networks*, pp. I/373-380

L. O. Chua, L. Yang (1988). Cellular neural networks: Theory; *IEEE Trans. Circuits Syst.*, Vol. 35, pp.1257-1272 and L. O. Chua, L. Yang (1988). Cellular neural networks: Applications; *IEEE Trans. Circuits Syst*., Vol. 35, pp.1273-1290

L. O. Chua, T. Roska (1993) The CNN Paradigm; *IEEE Trans. Circuits Syst.*, Vol. 40, No.3, pp. 147-156

J. Daugman (1988). Complete Descrete 2-D Gabor Transform by Neural Networks for Image Analysis and Compression; *IEEE Transactions on Acoustics, Speech and Signal Processing* Vol 36, No 7, pp. 1169-1179

P. Földiák (1989). Adaptive Network for Optimal Linear Feature Extraction; *IEEE Proc. Int. Conf. Neural Networks*; pp. I/401-405

Freisleben (1993). PCA in a Network with Full Lateral Connections; *Proc. Int. Conf. Art. Neural Networks ICANN-93*, Springer Verlag

K. Fukunaga (1972). **Introduction to Statistical Pattern Recognition**; Academic Press, New York

A. Habibi, P. Wintz (1971). Image Coding by Linear Transformation and Block Quantization; *IEEE Trans. on Comm. Techn.*, Vol. COM-19, No 1, pp.50-62

K. Hornik, C.-M. Kuan (1992). Convergence Analysis of Local Feature Extraction Algorithms, *Neural Networks*, Vol.5, pp.229-240

M. W. Hirsch (1989). Convergent Activation Dynamics in Continuous Time Networks; *Neural Networks*, Vol.2, pp. 331-349

JPEG-9-R6, International Organization for Standardization (ISO), 1991. Digital Compression and Coding of Χοντινυουσ-tone Still Images; in: JPEG-9-R6: Working Draft for Development of JPEG CD 10918-1.

MPEG 91, International Organization for Standardization (ISO), 1991. Coded Representation of Picture and Audio Information; MPEG Video Report, ISO-IEC/JTC1/SC2/WG11

A. K. Jain (1989). **Fundamentals of digital image processing**, Prentice Hall

N. S. Jayant, Peter Noll (1984). **Digital Coding of waveforms**, Prentice Hall .

J. Jones, L. Palmer (1987). The Two-dimensional Spatial Structure of Simple Receptive Fields in Cat Striate Cortex; *J.Neurophys.*, Vol.58,No 6 , pp. 1187-1211

T. Kohonen, E.Oja (1976). Fast Adaptive Formation of Orthogonalizing Filters and Associative Memory in Recurrent Networks of Neuron-Like Elements; *Biological Cybernetics*, *21*, pp.85-95

T. Kohonen (1982). Self-Organized formation of topologically correct feature maps, *Biological Cybernetics*, Vol 43, pp.59-69,

T. Kohonen (1988a). The "neural" Phonetic Typewriter of Helsinki University of Technology, *IEEE Computer*, March 1988

T. Kohonen (1988b). Representation of sensory information in self-organizing feature maps, and the relation of these maps to distributed memory networks, in: R. Cotterill (ed.), *Computer simulation in Brain Science*, Cambridge Univ. press, 1988

T. Kohonen (1989). Self-Organization and associative memory; Springer Verlag Berlin 1989

T. Kohonen (1993). Physiological Interpretation of the Self-Organizing Map Algorithm; *Neural Networks*, Vol.6, pp. 895-905

H.P. Kramer, M.V. Mathews (1956). A Linear Coding for Transmitting a Set of Correlated Signals; *Transact. 1956 Symp. on Inf. Theory*; in IEE Trans. on Inf. Th. IT-2 .

S. Kuffler, J. Nicholls, R. Martin: From Neuron to Brain; Sinauer Ass. Inc. Publ., Sunderland, MA 1984

T.K. Leen (1991). Dynamics of learning in linear feature-discovery networks; Networks, Vol. 2, pp. 85-105

Malsburg, Ch. v.d. (1973) Self-organization of orientation sensitive cells in the striate cortex, *Kybernetik*, Vol.14, pp.85-100

C.M. Marcus, F.R. Waugh, R.M. Westervelt (1991). Connection Topology and Dynamics in Lateral Inhibition Networks; in: R. Lippmann, J. Moody, D. Touretzky: **Advances in Neural Information Processing Systems 3,** Morgan Kaufmann Publ., San Mateo

Erkki Oja (1982). A Simplified Neuron Model as a Principal Component Analyzer,

J. Math. Biol. 13: 267-273

Erkki Oja (1989). Neural Networks, Principal Components, and subspaces, *Int. J. Neural Systems*, Vol 1/1 pp. 61-68

E. Oja (1991) Learning in non-linear Constrained Hebbian Networks; Proc. ICANN 91, T.Kohonen et al. (Eds.), **Artif. Neural Netw.**, Elsevier Sc. Publ., pp. 385-390

E. Oja, Ogawa, Wangviwattana (1992). Principal Component Analysis by Homogeneous Neural Networks, Part I: The Weighted Subspace Criterion, and Part II: Analysis and Extensions of the Learning Algorithm; *IEICE Trans. Inf. & Syst*, Vol. E75-D, No.3, May 1992, pp. 366-375, 376-382

K. Okajima (1986). A Mathematical Model of the Primary Visual Cortex and Hypercolumn; *Biol. Cyb.* Vol. 54, pp. 107-114

E. Pfaffelhuber, P.S. Damle (1973). Learning and Imprinting in Stationary and Non-Stationary Environment; Kybernetik Vol. 13, pp. 229-237

M. Plumbley (1993). Efficient Information Transfer and Anti-Hebbian Neural Networks; *Neural Networks*, Vol.6, pp.823-833

T. Roska (1993). The analogic single-chip CCN visual supercomputer - a review; in: Chetverikov, Kropatsch (Eds.), Computer Analysis of Images and Patterns, Springer Verlag , pp. 813-821

J. Rubner, P. Tavan (1989). A Self-Organizing Network for Principal-Component Analysis, *Europhys. Lett.*, 10(7), pp. 693-698 .

T. Sanger (1989). Optimal unsupervised Learning in a Single-Layer Linear Feedforward Neural Network; *Neural Networks* Vol. 2, pp.459-473

TCS93 (1993). Special issue on cellular neural networks, *IEEE Transactions on Circuits and Systems I*, Vol. 40, No. 3, March 1993

J. Tou, R. Gonzalez (1974). *Pattern Recognition Principles;* Addison Wesley Publ., Reading, MA

R. Williams (1985) Feature Discovery through Error-Correction Learning; *ICS Report 850*1, University of Cal. and San Diego

D. Willshaw, C. von der Malsburg (1976). How patterned connections can be set up by self-organization. Proc.Royal Soc. London, 194, 431-445

P. Wintz (1972). Transform Picture Coding; *Proc. IEEE*, Vol. 60, No. 7, pp. 809-820

NOMENCLATURE

$\mathbf{A}^T$	transpose of vector or matrix **A**
$\mathbf{x}(t)$	*t*th input data vector
$\mathbf{C}_{xx}$	autocorrelation matrix of *x*
z_i	activity of neuron *i*
T_i	lateral influence on neuron *i*
y_i	output of neuron *i*
$\mathbf{y}$	vector of outputs
$\tilde{\mathbf{w}}$	unnormalized weight vector
$\mathbf{w}_i$	normalized feed-forward weight vector of *i*th neuron
$\mathbf{W}$	matrix with weight vectors as rows
u_{ij}	lateral weight from neuron *i* to neuron *j*
$\mathbf{U}$	matrix of lateral weights
$\mathbf{B}$	matrix of all weights of the network
$\mathbf{I}$	identity matrix
$\mathbf{e}_i$	*i*th eigenvector of $\mathbf{C}_{xx}$
λ_i	*i*th eigenvalue of $\mathbf{C}_{xx}$
$\lambda_{min}(\lambda_{max})$	minimal (maximal) eigenvalue of $\mathbf{C}_{xx}$
$\mathbf{p}_{ij}$	subspace (plane) spanned by eigenvectors *i* and *j*
R(.)	risk function, objective function
S(.)	output (activation or squashing) function
L(.)	Lagrange function
R	lateral inhibition radius of a neuron
n	number of input dimensions
m	number of neurons in the net
N	length of picture
M	width of picture
c	two-dim. pixel data, ordered in vector form
K	number of subpictures in the picture
β	cross-correlation/autocorrelation ratio parameter
γ	proportional factor in learning equations (*learning rate*)

APPENDIX A: THE EXTREMES OF THE OBJECTIVE FUNCTION

Theorem:

The objective function

$$R(\mathbf{b}_1,\ldots,\mathbf{b}_m) = 1/4\ \beta \Sigma_i \Sigma_{j\neq i} (\langle y_i y_j\rangle)^2 - 1/2\ \Sigma_i \langle y_i^2\rangle$$

with the condition $|\mathbf{b}_k| = 1$ has as necessary condition for an extremum $R(\mathbf{b}_1^*,\ldots,\mathbf{b}_m^*)$ that the $\mathbf{b}_1^*,\ldots,\mathbf{b}_m^*$ are eigenvectors of the autocorrelation matrix $\mathbf{C}$.

Proof:

The constrained extremes of the objective function can be obtained by the method of Lagrange multipliers. Here, we construct the Lagrange function

$$L(\mathbf{b}_1,\ldots,\mathbf{b}_m, \mu_1,\ldots,\mu_m) = R(\mathbf{b}_1,\ldots,\mathbf{b}_m) + \mu_1(|\mathbf{b}_1|^2-1) + \ldots + \mu_m(|\mathbf{b}_m|^2-1)$$

The *2m* necessary conditions characterize the multivariate extremes

$$\frac{\partial L}{\partial \mathbf{b}_k} = 0 \quad \frac{\partial L}{\partial \mu_k} = 0 \qquad \forall k = 1, \ldots, m$$

and give us beside our *m* restrictions $|\mathbf{b}_k| = 1$ the *m* conditions

$$\nabla_b L(\mathbf{b}_k^*) = \nabla_b R(\mathbf{b}_k^*) + \mu_k \nabla_b(|\mathbf{b}_k^*|^2-1) = 0 \qquad \forall k = 1, \ldots, m \quad (A.1)$$

using the Nabla operator $\nabla_b F(\mathbf{b}) = (\partial F(\mathbf{b})/\partial b_1, \ldots. ,\partial F(\mathbf{b})/\partial b_n)^T$.

Let us evaluate the gradient $\nabla_b R(\mathbf{b}_k)$ first.

With eqn(2.6) we have

$$\nabla_b R(\mathbf{b}_k) = 1/4\beta\ \nabla_b \Sigma_i \Sigma_{j\neq i} (\langle y_i y_j\rangle)^2 - 1/2 \nabla_b \Sigma_i \langle y_i^2\rangle$$

With a different ordering of the sum

$$\sum_i \sum_{j\neq i} \langle y_i y_j\rangle^2 = \sum_{i\neq k} \sum_{j\neq i} \langle y_i y_j\rangle^2 + \sum_{j\neq k} \langle y_k y_j\rangle^2 = \sum_{i\neq k} \sum_{j\neq i} \langle y_i y_j\rangle^2 + \sum_{j\neq k} \langle y_k y_j\rangle^2 + \sum_{i\neq k} \langle y_i y_k\rangle^2$$

and by eqn (2.1) only the terms containing y_k remain non–zero in $\nabla_b R(\mathbf{b}_k)$ and we get

$$\nabla_b R(\mathbf{b}_k) = \beta \Sigma_{j\neq k} \langle y_k y_j\rangle \nabla_b(\langle y_k y_j\rangle) - \langle y_k \nabla_b y_k\rangle$$

$$= \beta \Sigma_{j\neq k} \langle y_k y_j\rangle \langle \mathbf{x} y_j\rangle - \langle \mathbf{x} y_k\rangle \qquad (A.2)$$

$$= \beta \Sigma_{j\neq k} \langle \mathbf{x}\mathbf{x}^T \rangle \mathbf{b}_j (\mathbf{b}_j{}^T \langle \mathbf{x}\mathbf{x}^T \rangle \mathbf{b}_k) - \langle \mathbf{x}\mathbf{x}^T \rangle \mathbf{b}_k = \beta \Sigma_{j\neq k} \mathbf{C}\mathbf{b}_j \mathbf{b}_j{}^T \mathbf{C}\mathbf{b}_k - \mathbf{C}\mathbf{b}_k$$

$$= \left[\beta \mathbf{C} \left(\Sigma_{j\neq k} \mathbf{b}_j \mathbf{b}_j{}^T \right) \mathbf{C} - \mathbf{C} \right] \mathbf{b}_k \qquad \text{(A.3)}$$

The condition (A.1) becomes

$$\nabla_a L(\mathbf{b}_k{}^*) = \left[\beta \mathbf{C} \left(\Sigma_{j\neq k} \mathbf{b}_j{}^* \mathbf{b}_j{}^{*T} \right) \mathbf{C} - \mathbf{C} \right] \mathbf{b}_k{}^* + 2\mu_k \mathbf{b}_k{}^* = 0 \qquad \forall k = 1,\ldots,m$$

$$\text{or} \quad \left[\mathbf{C} - \beta \mathbf{C} \left(\Sigma_{j\neq k} \mathbf{b}_j{}^* \mathbf{b}_j{}^{*T} \right) \mathbf{C} \right] \mathbf{b}_k{}^* = \theta_k \mathbf{b}_k{}^* \qquad \theta_k = 2\mu_k \qquad \forall k = 1,\ldots,m \qquad \text{(A.4)}$$

This is an eigenvector eqn for the matrix [.]. It is easy to see that this has as solutions the *m* eigenvectors of $\mathbf{C}$: Suppose the $\mathbf{e}_k$ are all eigenvectors of $\mathbf{C}$ and we have s_i weight vectors $\mathbf{b}_j$ converged to eigenvector $\mathbf{e}_i$. Then (A.4) becomes

$$\left[\mathbf{C} - \beta \mathbf{C} \left(\Sigma_{j\neq k} \mathbf{b}_j{}^* \mathbf{b}_j{}^{*T} \right) \mathbf{C} \right] \mathbf{e}_k = \lambda_k \mathbf{e}_k - \beta \mathbf{C} \left(\Sigma_{j\neq k} \mathbf{b}_j{}^* \mathbf{b}_j{}^{*T} \right) \lambda_k \mathbf{e}_k = \lambda_k \mathbf{e}_k - \beta \mathbf{C} \left(\Sigma_i s_i \mathbf{e}_i \mathbf{e}_i{}^T \right) \lambda_k \mathbf{e}_k$$

$$= \lambda_k \mathbf{e}_k - \beta \mathbf{C} \lambda_k s_k \mathbf{e}_k = (\lambda_k - \beta \lambda_k{}^2 s_k) \mathbf{e}_k = \theta_k \mathbf{e}_k$$

The eigenvectors of $\mathbf{C}$ are also the eigenvectors of the matrix [.] and fulfill condition (A.4).

Now, note that the rank of a linear transformation (i.e., the number of independent base vectors of the projection space or the number of independent row vectors of the corresponding matrix), which is composed of several linear transforms (e.g. $\mathbf{G} = \mathbf{D}\mathbf{B}$) can not be bigger then the rank of any of its transforms. This is based on the fact that two linear dependent base vectors $\mathbf{b}_i$ and $\mathbf{b}_j$ of a space remain dependent after a linear transform (e.g. $\mathbf{0} = \mathbf{D}\times\mathbf{0} = \mathbf{D}(a\mathbf{b}_i + b\mathbf{b}_j) = a\mathbf{D}\mathbf{b}_i + b\mathbf{D}\mathbf{b}_j = a\mathbf{g}_i + b\mathbf{g}_j = \mathbf{0}$) and reduce the dimension of the projection space to the dimension of the input space, i.e., the rank of the first matrix. This is also true for the transpose of $\mathbf{G}$, for $\mathbf{B}^T\mathbf{D}^T$, and means that the rank of $\mathbf{G}$ is reduced to the minimum of both, $\mathbf{D}$ and $\mathbf{B}$. Since this is true for all pairs of matrices in a chain, it is also true for the whole chain.

For our problem, this means that the rank of the matrix [.] is the minimum of the ranks of $\mathbf{C}$ and $(\Sigma_{j\neq k} \mathbf{b}_j{}^* \mathbf{b}_j{}^{*T})$ and we have maximally as many eigenvectors of matrix [.] as eigenvectors of $\mathbf{C}$ exist. Therefore, the eigenvectors of $\mathbf{C}$ are the only solutions for the extremes of the objective function,

Q.E.D.

APPENDIX B: LEARNING AN EXPECTATION VALUE

Theorem:

By the learning rule

$$r(t) = r(t-1) - 1/t \left(r(t-1) - \alpha v(t)\right) \qquad \text{(B.1)}$$

at every time step t the parameter m represents the expectation value

$$r(t) = \langle \alpha v \rangle_t \qquad \text{(B.2)}$$

of the stationary random variable v.

Proof: By inspection:

For $t = 1$, we have $r(1) = \alpha v(1)$, independently of the initial value r(0). Now, to prove eqn (B.1) by complete induction, it suffices to show that (B.1) is also valid for an arbitrary t under the assumption that it holds for $t-1$.

For that purpose, let us assume that $r(t-1) = \alpha\langle v \rangle = \alpha\ [1/(t-1)] \sum_{k=1}^{t-1} v(k)$ holds. Then the new average becomes

$$\alpha\ [1/t] \sum_{k=1}^{t} v(k) = \alpha\ [v(t)/t] + \alpha\ [(t-1)/(t-1)t] \sum_{k=1}^{t-1} v(k)$$

$$= a\ [v(t)/t] + [1-1/t]\ r(t-1)$$

$$= r(t-1) - 1/t[-\alpha v(t) + r(t-1)] = r(t)\ .$$

Thus, after each learning step (B.1) the weight r represents the average of αv.

APPENDIX C: NON-CENTRED INPUT

Many networks (e.g., Oja et al. 1992; Sanger, 1989; Rubner & Tavan, 1989) assume that the pattern statistics are centred, i.e., the expected input $\langle \mathbf{x} \rangle$ is zero. Then the covariance matrix $\langle (\mathbf{x}-\langle \mathbf{x} \rangle)(\mathbf{x}-\langle \mathbf{x} \rangle)^T \rangle$ becomes the autocorrelation matrix $\mathbf{C}_{xx} = \langle \mathbf{x}\mathbf{x}^T \rangle$. For the latter case, the normalized Hebbian (or anti-Hebbian) rule, let the weight vector converge to an eigenvector of the autocorrelation matrix, implementing a PCA. If the patterns are not centred, we are in trouble – how can we learn the eigenvectors of the covariance matrix for the PCA ?

This can be overcome by the following approach. Let us redefine the input $\mathbf{x}^T = (x_1,\ldots,x_n) \rightarrow \tilde{\mathbf{x}}^T = (x_1,\ldots,x_n,1)$ by an additional, constant line $x_{n+1} = 1$. Then the corresponding input weight w_{n+1} of $\tilde{\mathbf{w}}^T = (\mathbf{w}^T,w_{n+1}) = (w_1,\ldots,w_n,w_{n+1})$ is learned by the anti-Hebbian rule

$$w_{n+1}(t+1) = w_{n+1}(t) - \gamma(t+1) x_{n+1}(t+1)\, y(t+1) \qquad \text{(C.1)}$$

For the decreasing learning rate $\gamma(t) := 1/t$ and the output $y(t+1) = \tilde{\mathbf{w}}^T(t)\, \tilde{\mathbf{x}}(t+1) = \mathbf{w}^T(t)\mathbf{x}(t+1) + x_{n+1}w_{n+1}$ this becomes with the definition $z = \mathbf{w}^T\mathbf{x}$

$$w_{n+1}(t+1) = w_{n+1}(t) - 1/(t+1)\,\big(z(t+1) + w_{n+1}(t)\big) \qquad \text{(C.2)}$$

Replacing in eqn (C.2) literally $z(t+1)$ by v and w_{n+1} by r makes the learning rule (C.2) become the learning rule (B.1) with $\alpha = -1$. For this rule Appendix B proved that it learns the average $\alpha\langle z(t) \rangle$ at each time step. Thus, by the additional weight the output becomes $y = z - \langle z \rangle$ with the mean value $\langle y \rangle = \langle z - \langle z \rangle \rangle = 0$.

Please note that the time scale of this iteration must be smaller than the one we use for learning the other weights by the learning rules of eqn (2.11)-(2.14) in the neuron. This is necessary because we use the output y in the Hebb-type learning rules itself, demanding centred input, i.e., centred output for PCA convergence goals. Thus, to implement a PCA, the offset weight w_{n+1} must converge much faster than the other weights to ensure the convergence of the weights to the eigenvectors of the cross-correlation matrix, not to the ones of the autocorrelation matrix.

It can be shown (Brause, 1992b) that the autocorrelation matrix of the augmented input has the same eigenvectors as the covariance matrix of the non-augmented input.

APPENDIX D: THE ITERATION OF THE WEIGHT VECTOR

One iteration step of the learning rule is by eqns. (2.9), (2.10) and (2.12)

$$\mathbf{w}_i(t+1) = \frac{\mathbf{w}_i(t) + \Delta\mathbf{w}_i}{|\mathbf{w}_i(t) + \Delta\mathbf{w}_i|} = \frac{1}{g}\left[\mathbf{w}_i(t) + \gamma(t)\left(\mathbf{C}\mathbf{w}_i - \beta\mathbf{C}(\Sigma_{j\neq i}\mathbf{w}_j\mathbf{w}_j^T)\mathbf{C}\mathbf{w}_i\right)\right]$$

with $g = |\mathbf{w}_i(t) + \Delta\mathbf{w}_i|$.

Let us write this in the base of the orthonormal eigenvectors $\mathbf{e}_1, \ldots, \mathbf{e}_n$ of $\mathbf{C}$ by using the coefficients a_{il} in this base which are given by the identity

$$\mathbf{w}_i(t) = \Sigma_l\, a_{il}(t)\, \mathbf{e}_l$$

Then, the *k*th component, in the direction of $\mathbf{e}_k$, denoted also by indexed brackets $[.]_k$, evolves to

$$a_{ik}(t+1) = \left\{ a_{ik}(t) + \gamma\left(\left[\mathbf{C}\,(\Sigma_l\, a_{il}(t)\, \mathbf{e}_l)\right]_k - \left[\beta\mathbf{C}(\Sigma_{j\neq i}\mathbf{w}_j\mathbf{w}_j^T)\mathbf{C}\,(\Sigma_l\, a_{il}(t)\, \mathbf{e}_l)\right]_k \right) \right\} / g$$

$$= \left\{ a_{ik}(t) + \gamma\lambda_k\, a_{ik}(t) - \gamma\left[\beta\mathbf{C}\,\Sigma_{j\neq i}\mathbf{w}_j\mathbf{w}_j^T\,(\Sigma_l\, a_{il}(t)\lambda_l\mathbf{e}_l)\right]_k \right\} / g$$

$$= \left\{ a_{ik}(t) + \gamma\lambda_k\, a_{ik}(t) - \gamma\left[\beta\mathbf{C}\,\Sigma_{j\neq i}(\Sigma_l\, a_{jl}(t)\, \mathbf{e}_l)(\Sigma_l\, a_{jl}(t)\, \mathbf{e}_l)^T\,(\Sigma_l\, a_{il}(t)\lambda_l\mathbf{e}_l)\right]_k \right\} / g$$

$$= \left\{ a_{ik}(t) + \gamma\lambda_k\, a_{ik}(t) - \gamma\left[\beta\mathbf{C}\,\Sigma_{j\neq i}(\Sigma_l\, a_{jl}(t)\, \mathbf{e}_l)(\Sigma_l\, a_{jl}(t)\, a_{il}(t)\lambda_l)\right]_k \right\} / g$$

With the abbreviation $b_{ij} := \Sigma_l\, a_{jl}(t)\, a_{il}(t)\lambda_l$, concluded by the definition we have

$$\left[\Sigma_l\, a_{jl}\mathbf{e}_l\right]_k = a_{jk}$$

and finally get

$$a_{ik}(t+1) = \left\{ a_{ik}(t) + \gamma\lambda_k\, a_{ik}(t) - \gamma\left[\beta\mathbf{C}\,\Sigma_{j\neq i}b_{ij}\,\Sigma_l\, a_{jl}\mathbf{e}_l\right]_k \right\} / g$$

$$= \left\{ a_{ik}(t) + \gamma\lambda_k\, a_{ik}(t) - \gamma\beta\,\Sigma_{j\neq i}\, b_{ij}\,\Sigma_l\, a_{jl}(t)\left[\mathbf{C}\mathbf{e}_l\right]_k \right\} / g$$

$$= \left\{ a_{ik}(t) + \gamma\lambda_k\, a_{ik}(t) - \gamma\lambda_k\,\beta\Sigma_{j\neq i}\, a_{jk}(t)b_{ij} \right\} / g \qquad \text{(D.1)}$$

or

$$a_{ik}(t+1) = a_{ik}(t)\left\{ 1 + \gamma\lambda_k\,(1 - \beta\Sigma_{j\neq i}\, a_{jk}(t)b_{ij}/a_{ik}(t)\,) \right\} / g\ . \qquad \text{(D.2)}$$