TR－A－0150

Unsupervised Learning of Receptive Field Families on Regular Grids

Reiner Lenz

1992. 9. 14

ATR視聴覚機構研究所

〒619-02 京都府相楽郡精華町光台2-2 ☎07749-5-1411

ATR Auditory and Visual Perception Research Laboratories

2-2, Hikaridai, Seika-cho, Soraku-gun, Kyoto 619-02 Japan

Telephone: +81-7749-5-1411
Facsimile: +81-7749-5-1408

Unsupervised Learning of Receptive Field Families on Regular Grids

Reiner Lenz*
Cognitive Processes Department
Advanced Telecommunications Research Institute
Auditory and Visual Perception Laboratory
2-2 Hikaridai
Seika-cho Soraku-gun
Kyoto 619-02
JAPAN
Mats Österberg
Image Processing Laboratory,
Department of Electrical Engineering
Linköping University
S-59041 Linköping
Sweden

Abstract

We investigate families of receptive fields (i.e. low-level filter systems) that receive their inputs from sensors located on a finite, regular grid. We will first introduce a class of models that describe the behaviour of such systems. We introduce the representation theory of the dihedral groups to derive some important properties of such systems that originate in the structure of the grid (and not in the particular nature of the system). We will show that representation theory leads directly to algorithms with a structure similar to those of the FFT. We demonstrate possible applications of the theory in the field of low-level vision by showing how to construct and analyse different type of filter families. We will show that the same, universal, coordinate transformation simplifies all these different approaches.

In the second part of the paper we will discuss learning rules that lead to systems that learn these filter systems from examples. We will introduce three different types of systems: The Karhunen-Loeve, the quadratic and the fourth-order learning system. All these systems have the same structure that allows them to learn in parallel. The KL-system stablizes in states that are linear combinations of the eigenvectors of the input process. We will then introduce

*On leave from the Image Processing Laboratory, Department of Electrical Engineering, Linköping University, Linköping, Sweden

a new variation of the learning rule, based on second order terms that can differentiate between eigenvectors that belong to different eigenvalues. Finally we will introduce an energy function that contains fourth order terms. The resulting systems can no longer be analysed in terms of the covariance function of the input process but we will demonstrate empirically that they have a number of advantages over the ordinary KL-transform based systems. We will also show that systems that use the group theoretically defined coordinate transformation as pre-processing procedure perform better than the systems that work on the original pixel data.

1 Introduction

The investigation of receptive fields families (or early vision filters) is one of the basic problems in vision research and it has been investigated from a large number of viewpoints ([8], [9], [5], [6], [3], [16], [17], [24], [13], [12], [11], [29]). These filters are used to extract information from the data measured by the sensors of vision systems (like cameras or the retina). In their papers [13], [12], [11] Koenderinck and van Doorn note that most of these filters were constructed in a rather ad hoc manner and that they "constitute an odd lot, with little inherent coherence". They therefore propose a principled taxonomy of linear spatial receptive field families based on some general principles (see also [15] and [16] for related approaches).

Their approach is basically motivated by ideas from the scale space theory of image processing and the receptive field families they finally come up with are the Hermite (in cartesian coordinates) and the Laguerre functions (in polar coordinates) familiar from quantum mechanics and optics.

Although this result is very elegant from a theoretical point of view there are still some important open problems:

1. Using Hermite or Laguerre functions always implies an aperture function of infinite size since these functions have infinite support. Although they fall off exponentially fast it is well known that the introduction of binary windows of finite width complicates the study of these systems considerably. In optics these are the aberrations produced by finite apertures.
2. The model is continuous, i. e. all the functions used in the model are defined in regions with an infinite number of points. In reality we have however only a finite set of measurements (pixels) to work with. The problem of how to bridge the gap between the continuous model and the discrete data set requires a careful investigation (see for example the discussion on mathematical and physical operators in [31] or the literature on sampling theory).
3. The filter functions are designed (in this case in the framework of scale space theory). It is however not explained how (and why) these systems could evolve in natural systems.

These are some of the open problems connected with the approach described by Koenderinck and van Doorn and with continuous models in general. In this paper we will avoid these problems by using a discrete model instead. We observe that there are only finitely many sensors in the receptive fields and that they are usually arranged in a highly symmetrical way. This is

always true when the images are collected by a camera. The pixels represent, in this case, data on a square grid. In this paper we will mainly use the square grid but all results can be generalized to grid geometries where the locations of the sensors are the corners of regular polygons. Other interesting sensor geometries include images on hexagonal grids.

In the first part of the paper we will show how the representation theory of the dihedral groups can be used to analyze filter families. We show that for a receptive field of a given size and a given geometry there is one coordinate transformation under which all filter functions have an especially simple form. We show that this coordinate transformation has properties similar to the FFT. We will especially consider correlation matrices of input processes with group theoretically defined symmetry properties. For these processes we will show that the proposed coordinate transformation leads to a decorrelation of the transformed signals. The correlation matrix of the transformed signals will have a block-diagonal structures.

In the second part of the paper we describe several systems that can learn the eigenvectors of a correlation matrix from examples. The learning rules used by the systems make it possible to learn in parallel. From the results derived in the first part of the paper we know that the eigenvector analysis of a large matrix can be reduced to a number of eigenvector computations on smaller matrices if we consider the signal in the group theoretically derived coordinate system. This should speed up the learning process considerably since part of the eigenvector calculations are already hard-coded in the coordinate transformation.

In the last section we will demonstrate some of the properties of these filter systems with some examples. Some of the basic facts and results from the theory of group representations are summarized in the appendix A. The different learning rules are derived in appendix B.

2 The basic model

We will first introduce some notations and a general framework which will be used afterwards. We try to select a minimal set of constraints under which interesting results can be derived. First we introduce some notations:

A1 *The receptive field (i.e. the input of an early vision filter) consists of a finite number of sensor measurements.*

The number of measurements is denoted by N and the individual measurements by s_k. All input values are collected in the signal vector:

$$s = (s_1, \quad s_2, \quad ..., \quad s_N)' \tag{1}$$

The next assumption states that we consider systems of linear filters that analyze the same input data simultaneously.

A2 *The system can be described by a matrix of (complex) coefficients.*

This system matrix is denoted by A. The vector belonging to the k-th filter function is:

$$a_k = (a_{k1}, \quad a_{k2}, \quad ..., \quad a_{kN}). \tag{2}$$

The output o_k of the filter k given the signal vector s is the scalar product between a_k and s :

$$o_k = a_k \cdot s = \sum_{l=1}^{N} a_{kl} s_l \tag{3}$$

or in matrix notation:

$$o = A \cdot s \tag{4}$$

The number of linear filters in the system will usually be denoted by K.

From these two assumptions we conclude that there are at most N linearly independent filter functions. Such a maximal filter system will only perform a remapping of the original input data without any information loss. In general we are, of course, interested in filter systems with fewer number of filter functions. We also note that continuous models usually do not provide such an upper bound.

Next we assume that all filter vectors have the same length: $\|a_k\|^2 = \sum_l a_{kl}^2 = 1$. This does not restrict the generality of our approach since scaled versions of the filter would only produce scaled output values but they would not add any new information.

A3 *Filter vectors are normed*

The next assumption is rather general and we will illustrate it with a number of examples. Different realizations of this principle gives us diffrent types of models:

A4 *We assume that there is a matrix C that governs the properties of the filter system.*

Usually we will be able to define the "best" filter system in terms of the eigenvectors and eigenvalues of the matrix C. This makes these models especially attractive in the context of learning since we can "learn" the best systems iteratively by using optimization methods. The investigation of several learning rules will be the topic of the second part of our paper.

Here are some examples that illustrate the assumptions made so far:

1. Assume the input signals s are generated by a stochastic process and that the system consists of one filter function. We define the best system as the system with the maximum mean squared response to the input signals. The best system is thus described by the vector $\hat{a}$ with:

$$\text{mean}_s |\langle a, s\rangle|^2 \leq \text{mean}_s |\langle \hat{a}, s\rangle|^2 \tag{5}$$

Computing the mean squared response of a filter vector a gives:

$$\begin{aligned} \text{mean}_s |\langle a, s\rangle|^2 &= \text{mean}_s \left(\langle a, s\rangle \langle s, a\rangle\right) = \text{mean}_s \left(a' s s' \overline{a}\right) = \\ &= a' \left(\text{mean}_s \left(ss'\right)\right) \overline{a} = \langle a, Ca\rangle \end{aligned} \tag{6}$$

where C is the covariance matrix of the input process. The best filter vector is the eigenvector $\hat{a}$ of C with the largest eigenvalue.

The matrix C is in this case the covariance matrix of the input process and the extremum principle we use is:

$$\langle a, Ca\rangle \leq \langle \hat{a}, C\hat{a}\rangle \tag{7}$$

2. The matrix C in the last example was directly coupled to the filter coefficients a. In the more general model we have to consider a whole family of filter vectors simultaneously. This makes it necessary to introduce a new measurement of the quality of a filter system. We proposed a quality function based on properties of the output vectors. The output vectors computed from all possible input signals form a cloud in $K-$dimensional feature space. We found it useful to require that this cloud should occupy as large a volume as possible in feature space. We measured the size of the cloud by the determinant of the covariance matrix of the output vectors. This property leads to filter systems A for which

$$\det C = \det\left(c_{kl}\right) = \det\left(\text{mean}(o_k o_l)\right) \tag{8}$$

is maximum.

In this model the matrix C is defined in terms of the mean values of the products of feature values. The feature values are in turn scalar products between the input signals and the filter vectors.

A careful analysis of this system shows that the best filter systems consist of the eigenvectors of the covariance matrix of the input process with the largest eigenvalues or of linear combinations of these eigenvectors. These facts are proved in [23].

3. Next we use a mechanical, membrane-type model for the filter functions on the receptive field. We assume that the filter value at a location on the grid represents the distance of a particle from an equilibrium state. We also assume that neighboring particles are connected by springs and that each particle has a spring that pulls it back to the equilibrium state. The whole system follows the laws of classical mechanics (one can also think of the filter as a vibrating membrane). Finally we impose the boundary condition that the filter values at the boundary have value zero.

   This model ranks the filter kernels according to their potential energy. The kernels with extremal values of potential energy are the eigenvectors of a matrix C that can be obtained by a discretization of the motion equation. The size of C is $N \times N$ where N is the number of pixels. We consider only one filter function and denote the filter coefficients by a_k. To calculate the force that acts on a given location we note that (in the case of a four-connected neighborhood) this force is a sum of five terms: the spring that pulls the particle back to equilibrium and the four forces from the four neighboring positions. If we denote the position of the particle at the center by a_c, the values at the neighboring locations by a_s, a_w, a_n and a_e (south, west, north and east) and the spring constants by c_c (for the center spring pulling the particle back to equilibrium) and c_n then we find that the force acting on the center pixel is given by:

$$a_c \cdot c_c + c_n \cdot ([a_s - a_c] + [a_w - a_c] + [a_n - a_c] + [a_e - a_c])$$

   which can be rewritten as:

$$a_c \cdot (c_c - 4 \cdot c_n) + c_n \cdot (a_s + a_w + a_n + a_e)$$

For an inner point this leads to a Laplacian type filter kernel of the type

$$\begin{pmatrix} 0 & c_n & 0 \\ c_n & c_c - 4 \cdot c_n & c_n \\ 0 & c_n & 0 \end{pmatrix}$$

Rewriting this in vector notation shows that the matrix C is a bandmatrix with the terms $c_c - 4 \cdot c_n$ in the main diagonal and c_n in four neighboring diagonals. Special attention is needed if the membrane is pinned down at the boundaries.

4. This mechanical model can be given another interpretation that might be more appealing for low-level vision problems: If the pixel value at the center is approximately equal to the pixel value at one of the neighborhood positions then there is a correlation between these two pixel values. The energy function can thus also be interpreted as a measure of how much correlation we have in the neighborhood of a given pixel. The detection of such correlations is certainly of greatest importance in low level image processing since these structures form the basic building blocks for all further processing.

5. In the next model we assume that the pixel value at a point represents the number of particles located at this point. We assume further that there are a large number of such particles and that they all perform some kind of random motion. In the simplest case this motion is completely isotropic: a given particle located at the origin selects with probability $\frac{1}{4}$ a given neighbor on the grid and then it moves with probability t to this neighboring location. If s_0, s_s, s_n, s_w, s_e are the number of particles (pixel values) at the origin and the south, north, west and east neighbor location then the expected number of particles at the origin (after some time step) is given by:

$$(1-t) \cdot s_0 + \frac{t}{4} \cdot (s_s + s_n + s_w + s_e).$$

The mean change of the number of particles Δ_0 at the origin is given by:

$$\begin{aligned} \Delta_0 &= s_0 - (1-t)s_0 - \frac{t}{4}(4 \cdot s_0 - s_s - s_n - s_w - s_e) \\ &= \frac{t}{4} \cdot (4 \cdot s_0 - s_s - s_n - s_w - s_e) \end{aligned} \tag{9}$$

This is the discrete version of the diffusion equation and it is described by the 2-D kernel

$$\begin{pmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{pmatrix}$$

The corresponding matrix C is again a bandmatrix with entries in five diagonals as in the case of the membrane model.

Our last assumption is based on the observation that the sensors are usually arranged in a quite orderly, symmetrical fashion: Images from cameras come in rows and columns, other sensors may be arranged on a polar grid etc.. We know furthermore that simple geometric operations (like discrete rotations and

transpositions) have no dramatic effect on the low-level processes considered here. (This does not hold for higher processing levels).

These heuristical ideas are formalized in the following construction (some of the basic facts about groups and representations are collected in the appendix, but for a detailed treatment the reader has to consult the literature [10], [25], [4], [17]):

We consider permutations that rearrange the pixels on the grid.

$$g : (1, ..., N) \mapsto (g(1), ..., g(N)) \tag{10}$$

Next we assume that we have a number (say L) of such transformations and we require that they form a group (see the appendix for a definition). The set of all permutations of N symbols forms a group (the so-called symmetrical group) and the transformations $G = \{g_1, ..., g_L\}$ are a subgroup of the symmetrical group. We call the group G the *symmetry group* of our model.

In this paper we will only consider special permutation subgroups that are linked to the geometry of the grid under consideration. These groups are the *dihedral groups* $\mathcal{D}_n$. $\mathcal{D}_n$ is defined as the group of all isometric mappings that leave a regular, n-sided polygon invariant. It consists of n-rotations and n-transpositions around the diagonals of the polygon. This group has thus 2n elements and it is not commutative. This means that we cannot exchange the order in which these transformations are applied: rotating first and transposing then is usually different from transposing first and rotating then. This non-commutative behavior has some very serious consequences and it complicates the study of these groups considerably. Of special interest for image processing is the group $\mathcal{D}_4$ which maps the square grid into itself. Hexagonal grids may also be of interest but it turns out that all dihedral groups can be treated in the same way.

Now consider one element g from the symmetry group and a function f on the receptive field. Since the receptive field has only finite elements we can write f as a vector $f = (f_1, \ ..., \ f_N)$. If we first apply the transformation g to the grid and if we then apply the function f then we get a new function f_g:

$$f = (f_1, \ ..., \ f_N) \mapsto f_g = (f_{g(1)}, \ ..., \ f_{g(N)}) \tag{11}$$

The mapping $f \mapsto f_g$ is a linear operator on the N-dimensional vector space of all functions on the receptive field and we can therefore describe it by a matrix $T(g)$:

$$f_g = T(g)f. \tag{12}$$

For each element g we have thus a matrix $T(g)$ and it can be easily seen that these matrices have the property:

$$T(g_1 g_2) = T(g_1)T(g_2) \tag{13}$$

for all elements g_1, g_2 in the symmetry group. We call the mapping T a *matrix representation* of the group G

We now use these transformation matrices $T(g)$ to impose some restrictions on the system matrix C introduced in assumption A4: We call the matrix C *$G-$symmetric* if the matrix equations:

$$T(g)C = CT(g) \tag{14}$$

hold for all elements g in the symmetry group. This is our last assumption:

A5 *The matrix C is G-symmetric.*

In the simplest case deals with images on a square grid. The transformations g are the $0°, 90°, 180°$ and $270°$ rotations and the transpositions on the x- and y-axis and the diagonals. These transformations are the only isometries that leave the square fixed. The symmetry group consists of eight elements and the matrices $T(g)$ are the permutation matrices defined in equations 11 and 12.

The main point here is that each group element g imposes a constraint (equation 14) on the matrix C. The larger the group G is the more constraints we put on the matrix C and the fewer matrices C will satisfy all these constraints. The matrices that pass all these tests have a number of special properties and the main goal of the theory of group representations is it to detect these properties and to describe them in different forms.

Now consider the example where the matrix C was the covariance matrix of a stochastic process. We say that the stochastic process is G-symmetric if all the signals $\{T(g)s : g \in G\}$ have the same probability. All the rotated and transposed versions of an input signal are then equally probable. It is easy to see that the covariance matrix is G-symmetric in the sense of equation 14.

The matrices C used in the membrane and in the diffusion model are all G-symmetric since they were constructed in a way that made them independent of the transformations g in the dihedral groups: In the membrane model the forces from all four directions were defined in the same way and also the diffusion process has no built-in preferences for one of the neighbors. The symmetry conditions can, of course, also be checked by brute-force calculations of the matrix products in equation 14.

3 Results from Representation Theory

The derivation of the results described in this section is far beyond the scope of this paper we will therefore only summarize the few important aspects of the complete theory that are of importance here. Some of the results are described in the appendix but the interested reader should consult the literature to get a detailed description (see for example [4] for a general investigation of finite groups, [25] for a description of the general theory, [7] for a detailed study of rotation and Lorentz groups and [10] and [17] for an introduction and some applications in image science).

The G-symmetric matrices C satisfy the matrix equations $T(g)C = CT(g)$ for all group elements g of the symmetry group G and the representation T. The main property of C that will be used in the following is a consequence of Schur's Lemma:

Theorem 1 Assume that C is a G-symmetric matrix: $T(g)C = CT(g)$. Then there is a matrix M such that the transformed matrix: $M'CM$ is a block diagonal matrix of the form:

$$\begin{pmatrix} C_1 & 0 & 0 & \dots & 0 \\ 0 & C_2 & 0 & \dots & 0 \\ 0 & \ddots & \cdot & \dots & 0 \\ 0 & 0 & 0 & \dots & C_L \end{pmatrix} \tag{15}$$

where the C_l are square matrices.

This fundamental result requires some additional comments (a more detailed description can be found in the appendix):

1. The main point is that *one* matrix M is sufficient to block-diagonalize *all* G-symmetric matrices C.
2. The structure (i.e. the size of the matrices C_l and the number L) is only a function of the group G and the mapping T.
3. Given the group G and the mapping T it is possible to compute the transformation matrix M automatically.
4. For the dihedral group $\mathcal{D}_n$ the entries of M are all n-th roots of unity. It is thus very similar to the DFT. For the square grid and the group $\mathcal{D}_4$ it contains only the numbers $1, -1, i$ and $-i$. It can therefore be calculated with only additions and subtractions. This should make it very attractive for hardware implementation.

In the following table we have collected some decompositions for a number of neighborhood sizes. This decomposition uses the $\mathcal{D}_4$ group. It should give a feeling for the simplifications achievable with this method.

			block sizes					
grid size	pixels	matrix size	C_1	C_2	C_3	C_4	C_5	C_6
2	4	4×4	1	0	0	1	1	1
3	9	9×9	3	0	1	1	2	2
4	16	16×16	3	1	1	3	4	4
5	25	25×25	6	1	3	3	6	6
6	36	36×36	6	3	3	6	9	9
7	49	49×49	10	3	6	6	12	12
8	64	64×64	10	6	6	10	16	16
9	81	81×81	15	6	10	10	20	20
10	100	100×100	15	10	10	15	25	25
11	121	121×121	21	10	15	15	30	30

From the table we see that the size of the matrices is reduced with a factor of about four. We also see that the last two matrices always have the same size. It can be shown that they are not only of the same size but that they are actually identical. In the last case of an 11×11 neighborhood the original problem involving a 121^2 matrix is thus reduced to five problems of size $21, 10, 15, 15$ and 30 respectively. Apart from the reduction of the number of computations involved the computations are also numerically more stable since only smaller matrices are involved. Finally we want to point out that the savings are even more significant if we study dynamical systems like discrete diffusion equations.

We close this section of a paper with some results involving the introductory examples:

- In the first example we consider a random pattern on an receptive field of size 11×11. From this prototype pattern we generate new patterns by applying all transformations of the $\mathcal{D}_4$ group to it. From the construction of the input signals we know that the correlation matrix of this process is $\mathcal{D}_4$ symmetric and that it will have a block-diagonal structure in the new coordinate system. The original correlation matrix C and

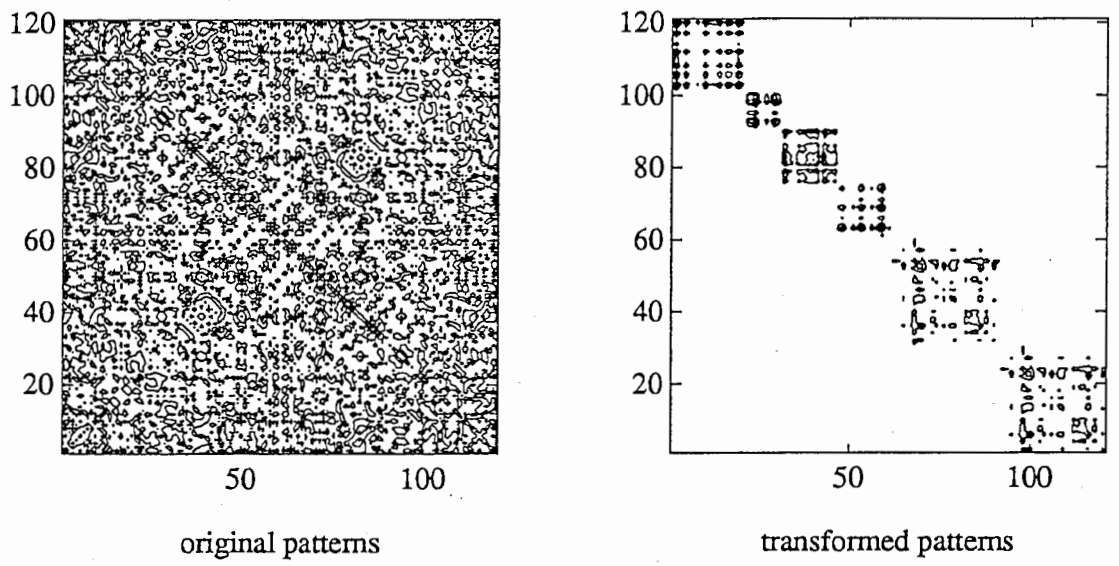

Figure 1: Correlation matrices for a symmetric pattern process

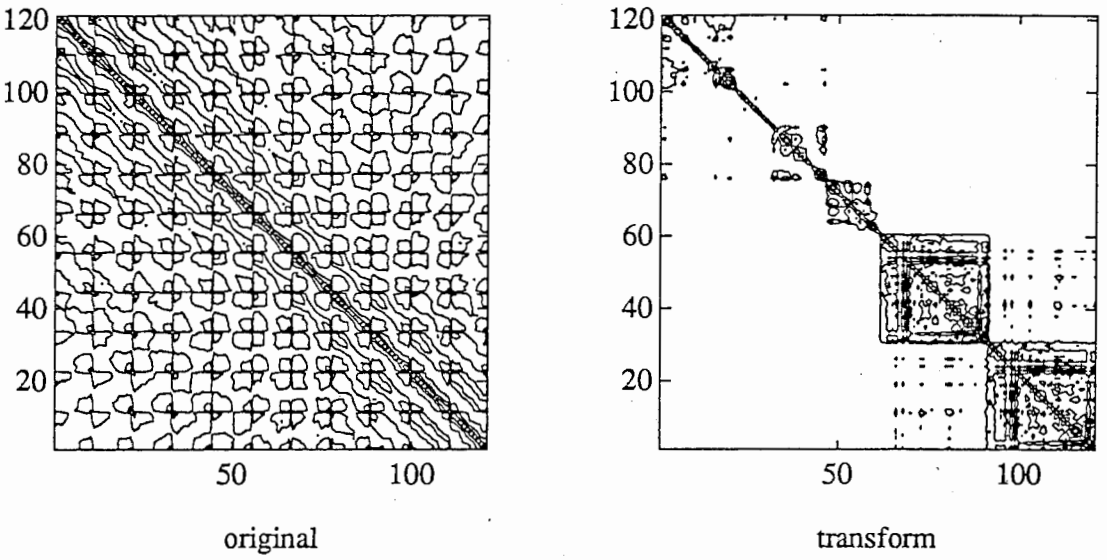

Figure 2: Correlation matrices for a randomly selected patterns

the correlation matrix $\overline{M}'CM$ in the new coordinate system are shown in figure 1:

- The previous example was constructed to fit the theory. In the next example we show that real images posses a dihedral symmetry. In this experiment we selected randomly three images from our database: Lenna, a satellite image of the moon and an image of an airport. From each of these images we selected randomly 500 neighborhoods of size 11 × 11. From these samples we computed the correlation matrix C and the transformed correlation matrix $M'CM$ (with the same matrix M as in the previous experiment). The result is shown in the next figure 2
- Computing the first three eigenfunctions of the discrete membrane (fixed at the border) with the smallest energy gives the filter functions shown in figure 3: They are blob- and edge-detectors.

In these introductory exposition of the theory we described only the simplest cases in which the group acted on the raw signal values s_k. The same algorithms work however also if the group acts on polynomials signal values or other transformations of the original signals.

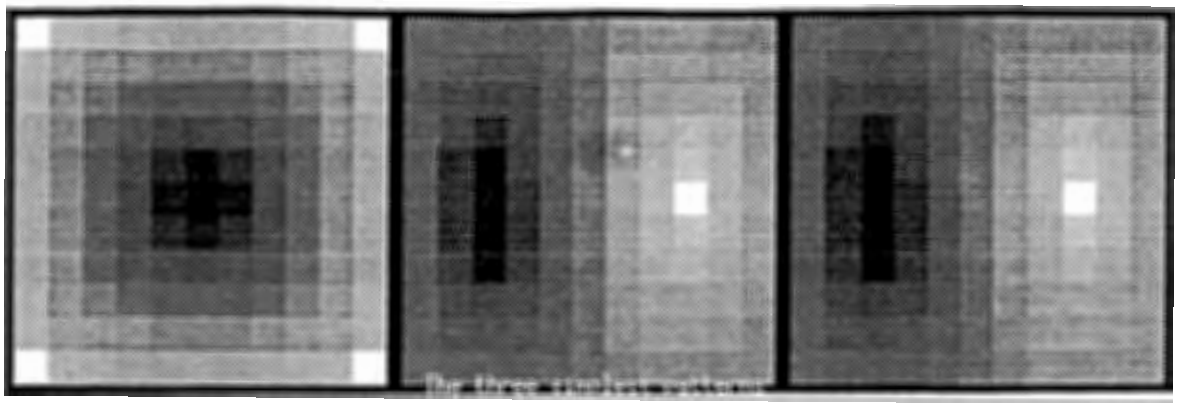

Figure 3: The three filters with minimal energy

4 Unsupervised Learning

The goal of unsupervised learning neural networks is it to discover structures in the space of input signals. One type of unsupervised linear neural networks is closely related to the traditional statistical method of principal component analysis (or Karhunen-Loeve expansion or eigenvalue analysis). One of the earliest attempts to compute one eigenfunction of a class of input signals is the principal component analyzer developed by Oja (see [26] and [27]). Oja's analyzer acts as a similarity detector or a correlation unit. This leads naturally to a Hebbian learning rule which updates the coefficients of the system so that the output of the analyzer is maximized. Oja showed that this analyzer could learn the first eigenfunction of the input process. Sanger (see [30]) generalized Oja's result by showing how different one-dimensional analyzers could be used to compute a number of different eigenfunctions. The system proposed by Sanger consists of a number of analyzers of the type introduced by Oja which are connected in a serial fashion. Another approach to generalize Oja's result was investigated by Leen in [14] where he uses lateral connections trained by an anti-Hebbian rule to penalize the interaction between different units. Leen shows how the analyzers can be forced to converge to different eigenfunctions in a parallel fashion.

In a series of papers ([19], [20], [18], [28], [21], [23], [22]) we have investigated parallel, feed-forward, one-layer linear networks without any lateral connections. Following Oja we use a linear correlator as the basic unit of our system. The complete system consists of a number of these basis units. The internal state of this correlator is described by a set of variables which we collect in a vector. We call this vector the state vector of the unit. The state vector of the $k-th$ basic unit at time t will be denoted by $a_k^{(t)} = (a_{k1}, a_{k2}, ...a_{kN})$. When it is clear (and we fear that also sometimes when it is not) we will drop the reference to the time variable t. At each time step the correlator computes the scalar product between the current input signal s and the internal state vector. The result is the output value of the unit: $o_k^{(t)} = \left\langle a_k^{(t)}, s \right\rangle$ Then it updates the internal state vector: $a_k^{(t)} \mapsto a_k^{(t+1)}$. We will also assume that the state vectors are real and have norm one. The dynamic behavior of is governed by an energy function since the update rule tries to move the system to a point with minimal energy. Different energy functions lead to different update rules and therefore to different systems. In the following we

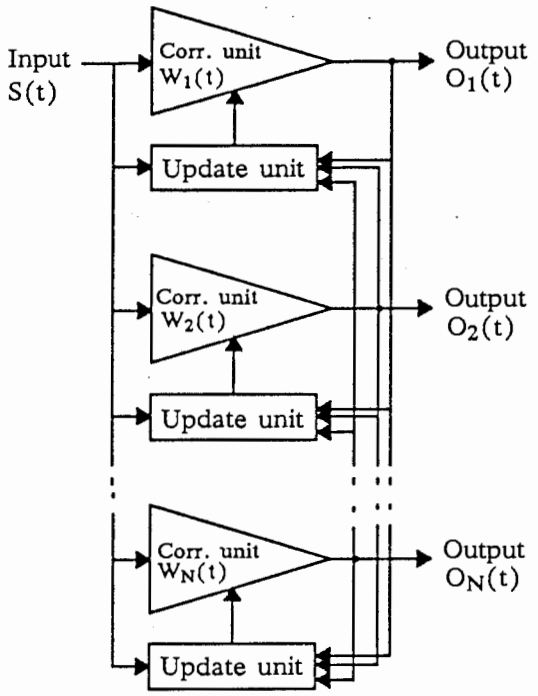

Figure 4: The Learning Filter System

will describe three such energy functions and the resulting systems. All these energy functions are defined in terms of the output values o_k alone. Computing the gradient of these functions it can then be very easily seen that this implies that no lateral connections are neeeded in such systems. All these systems can thus work in parallel and they all have the conceptional structure shown in figure 4. In the next section we will first introduce an energy function which leads to a system that learns the eigenfunctions of an input process. Then we will modify the system so that eigenfunctions belonging to different eigenvalues will be learned by different units. These energy functions are based on second order moments of the input process and we will therefore call them second order systems. We will then show that second order systems have some disadvantages and we will therefore introduce fourth order systems which avoid these problems. In the last section we will then describe some experiments that demonstrate the performance of these systems. Finally we note that instead of minimizing an energy function E we could also maximize a quality function Q. These two approaches are essentially equivalent. We will mainly use the quality function description in the following.

5 Three different learning rules

In the remaining sections of the paper we assume that the input signals are centered (have mean value zero) and normed (have length one). One basic idea behind all the systems discussed is that the system should extract as much "information" as possible from the incoming signals. We write "information" since we do not use the term in the ordinary, entropy definition sense. Instead we require that the cloud of extracted feature vectors should occupy as large section of the feature space as possible. This volume is measured by the

determinant of the covariance matrix of the feature vectors:

$$Q_V(A) = Q_V = \det([o_i o_j]) \stackrel{!}{=} \text{MAX} \tag{16}$$

Here and in the following we use the convention that square parenthesis denote averaging over all input signals: $[o_i o_j]$ is the second order moment of the feature vectors. Q_V depends only on the second order statistical moments of the input process since the basic units are all linear. Similar ideas, involving entropies, were recently discussed by Atick in [1].

In [23] we showed that the optimal filter functions derived from Q_V are orthonormal transformations of the eigenfunctions of the input covariance matrix. The system learns thus the Karhunen-Loeve transform.

Choosing the determinant in the quality function is quite natural since the volume measuring property of the determinant is characterizing the determinant (i.e. up to a constant the selection of the determinant is unique). From a practical point of view the determinant is however quite cumbersome since gradient based update rules lead to complicated, computation-intensive expressions. In the large number of experiments we made with this and similar learning rules we found that the determinant based methods are usually the only ones that always work as predicted without any additional parameter estimation procedure. This observation seems to confirm the unique role of the determinant and it usually compensates the difficulties originating in the difficult and slow update-rule.

If A is the matrix with the filter coefficients and C is the covariance matrix of the input process then we compute $Q_V = det(ACA')$. If the filter vectors in A are orthonormal then AA' is the unit matrix and for every orthogonal matrix T we can construct a new optimal solution TA of Q_V since the value of the determinant is invariant under orthogonal transformations. The function Q_V does therefore not uniquely determine the optimal solutions. In addition to the maximum volume principle we found it useful to require that the filter system should concentrate the output signals to as few output channels as possible. This should be a clear advantage in the case where the units have to communicate through noisy channels (as in the case of cells in living beings).

In our first approach to incorporate this idea into our model we introduce the quality function

$$Q_Q(A) = \frac{Q_V}{Q_2} \tag{17}$$

(where A is the filter coefficient matrix) with:

$$Q_2 = \sum_{i=1}^{N} \left[o_i^2\right] \left(1 - \left[o_i^2\right]\right) \tag{18}$$

This definition selects a filter system in which the correlation unit i should produce (in the mean) either a very strong or a very weak output signal. Q_2 is also a function of the second order moments of the training sequence. We call a system based on the quality function Q_V a Karhunen-Loéve Filter System and a system based on the quality function Q_Q a second order system. In [23] we showed that these second order systems can discriminate between eigenvectors belonging to different eigenvalues.

Karhunen-Loeve expansions are based on the covariance matrix of the input process. They are entirely based on averaged information all information

about individual patterns is lost. Two enterly different input processes have the same expansions if they posses the same first and second order moments.

This can be serious drawback as we can see in the next example:

For simplicity we consider signals that are functions on the unit circle. We select one such signal $s(x)$ and assume that all input signals are shifted versions of this signal: $s_\xi(x) = s(x - \xi)$ and that all these signals are equally probable.

The filter functions $a_k(x)$ compute the output values:

$$o_k(\xi) = \frac{1}{2\pi} \int_0^{2\pi} s(x - \xi)\overline{a_k(x)}\, dx \tag{19}$$

Next we expand the functions s and a_k into Fourier series:

$$s(x) = \sum \sigma_\nu e^{i\nu x}, \qquad a_k(x) = \sum \alpha_{k\mu} e^{i\mu x} \tag{20}$$

and compute the output values in terms of these Fourier coefficients:

$$o_k(\xi) = \frac{1}{2\pi} \int_0^{2\pi} \left(\sum_\nu \sigma_\nu e^{i\nu(x-\xi)} \right) \left(\sum_\mu \overline{\alpha_{k\mu}} e^{-i\mu x} \right) dx = \sum_\nu \sigma_\nu \overline{\alpha_{k\nu}} e^{-i\nu\xi}. \tag{21}$$

This gives finally for the correlation coefficients:

$$\begin{aligned} [o_k o_l] &= \frac{1}{2\pi} \int_0^{2\pi} o_k(\xi)\overline{o_l(\xi)} d\xi \\ &= \frac{1}{2\pi} \int_0^{2\pi} \sum_\nu \sigma_\nu \overline{\alpha_{k\nu}} e^{-i\nu\xi} \sum_\mu \overline{\sigma_\mu} \alpha_{l\mu} e^{i\mu\xi}\, d\xi \\ &= \sum_\nu \sigma_\nu \overline{\alpha_{k\nu}\sigma_\nu} \alpha_{l\nu} \\ &= \sum_\nu |\sigma_\nu|^2 \overline{\alpha_{k\nu}} \alpha_{l\nu} \end{aligned} \tag{22}$$

From this we see that the second order moments are only functions of the magnitudes of the Fourier coefficients of the original signal s. No information about the correlation between the different Fourier coefficients can be retrived for the second order moments.

From this result we conclude that the quality function should contain terms of order higher than two. We thus replace $[o_k^2] \cdot [(1 - o_k^2)]$ by $[o_k^2 \cdot (1 - o_k^2)] = [o_k^2] - [o_k^4]$. The complete quality function is now:

$$Q_{LFS}(A) = \frac{Q_V}{Q_4} \tag{23}$$

with:

$$Q_4 = \sum_{k=1}^{K} \left[o_k^2 - o_k^4 \right] \tag{24}$$

This quality function tries to learn filter functions that extract as much information as possible from the input signals and that tries to produce feature vectors with more or less binary components. A filter system based on this quality function is called a Learning Filter System or a fourth order system. In our experiments we found that this system had the best performance among all the investigated systems. Because of the fourth order term it is however very difficult to investigate the behavior of this system analytically.

A complete derivation of the learning rule in which we used a Newton based gradient search can be found in the appendix B. Here we only want to make some more comments on the form of the quality function and some variations of it.

The quality functions consists of two parts Q_Q and Q_4 which try to achieve *contrary* goals the Q_Q part tries to spread out the output signals as much as possible whereas the Q_4 part tries to concentrate them as much as possible. In the extreme case where all the filter outputs are zero we have an optimal solution for the Q_4 part but we have the worst value for Q_Q. In the usual approach to design energy functions with different components (say E_1 and E_2) one tries to combine them additively and to control the influence of both components with the selection of a weighting factor. The usual energy functions then take the form $E = E_1 + \lambda E_2$. This leads however to the problem of finding a "good" value for the weight factor. This is a highly non-trivial problem and the experimental adjustment of this factor is usually computationally expensive. We tried several variants of such additive functions but it always turned out that finding the "right" energy function was non-trivial. In the quality functions described in equations 17 and 23 there are no such free parameters and they usually work as expected.

6 Experimental results

We use first some artificially created data sets to demonstrate some of the properties of the different systems. The signals are one-dimensional since the results are easier to visualize in this case. Furthermore we can establish some of the results analytically.

In the first series of experiments we used as training sets one-dimensional vectors with 256 elements. The first data set (denoted by *cos(x), cos(3x)* in the plots) consisted of 256 shifted versions of cos(x) and 256 shifted versions of cos(3x). The second data set (denoted by *cos(x)+cos(3x)* in the plots) consisted of 256 shifted versions of the function $\gamma \cdot (\cos(x) + \cos(3x))$ where γ is a normalization constant. Three typical patterns from these data sets are shown in figure 5 . The patterns are all centered and the two pattern classes have the same correlation matrices as can be seen in figure 6 For each of the two data sets we trained the three diffferent systems with 2000 examples. The result is shown in the next series of figures: figures 7, 8 for the determinant based system, figures 9, 10 for the second order system and figures 11, 12 for the fourth order system. We see that the determinant based system mixes the four eigenfunctions cos(x), sin(x), cos(3x) and sin(3x) in a random fashion. The second and fourth order systems try to separate these four functions and the difference between the fourth order filter functions from data set one and data set two is greater than the differences between the corresponding filter functions learned with the second order system.

In the next series of figures we demonstrate the sensitiveness to differences in the higher order statistics of a signal set. We used two data sets based on square waves with period one and two and arbitrary location of the jump. These signals could represent the angular distribution of step edge and line patterns in a polar coordinate system. In the first data set (denoted by *edge,lines* in the figures) we used as input signals shifted versions of the pure period one and periode two patterns. In the second data set (denoted

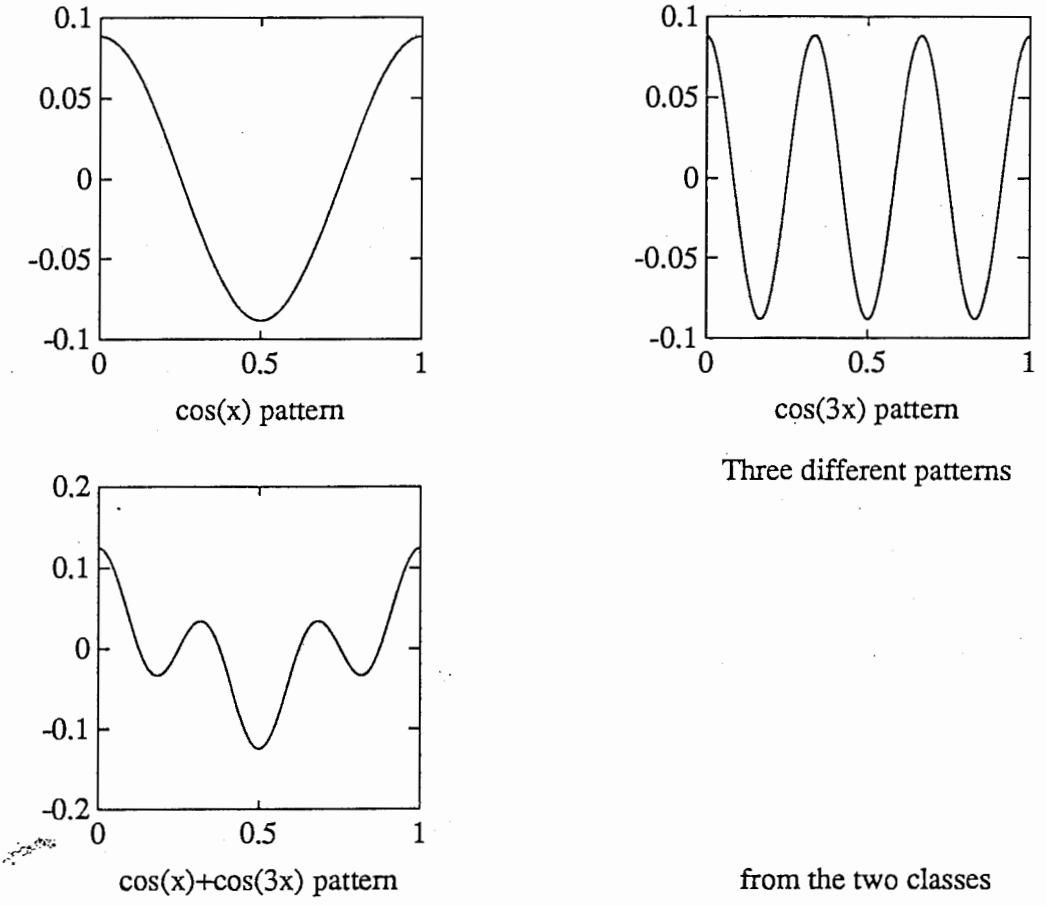

Figure 5: Three different signals from two different classes

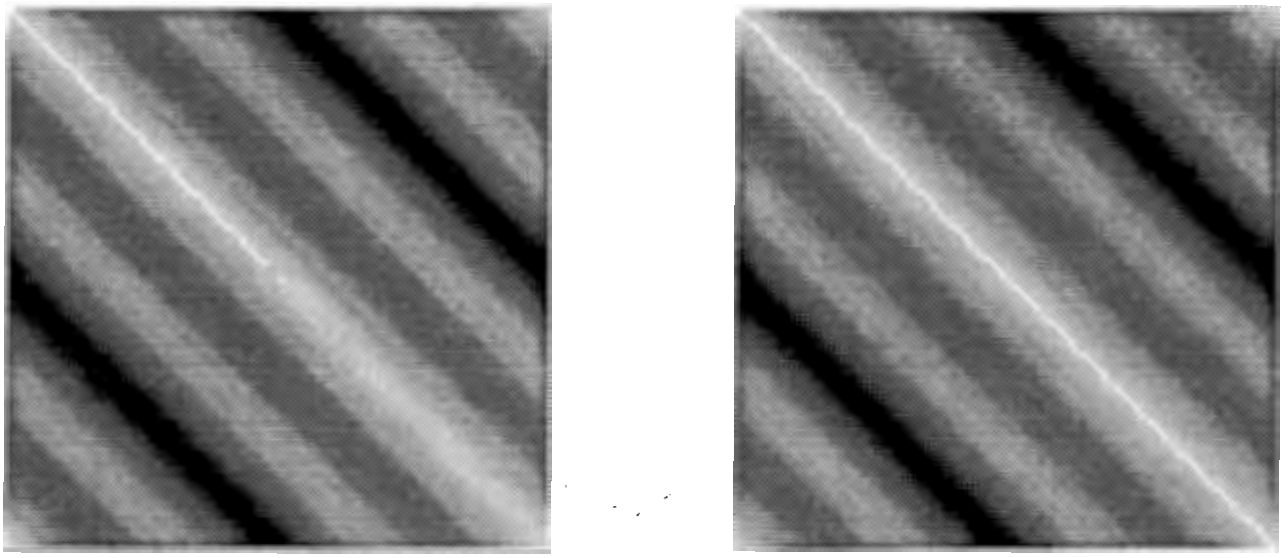

Figure 6: The correlation matrices of the two different sequences

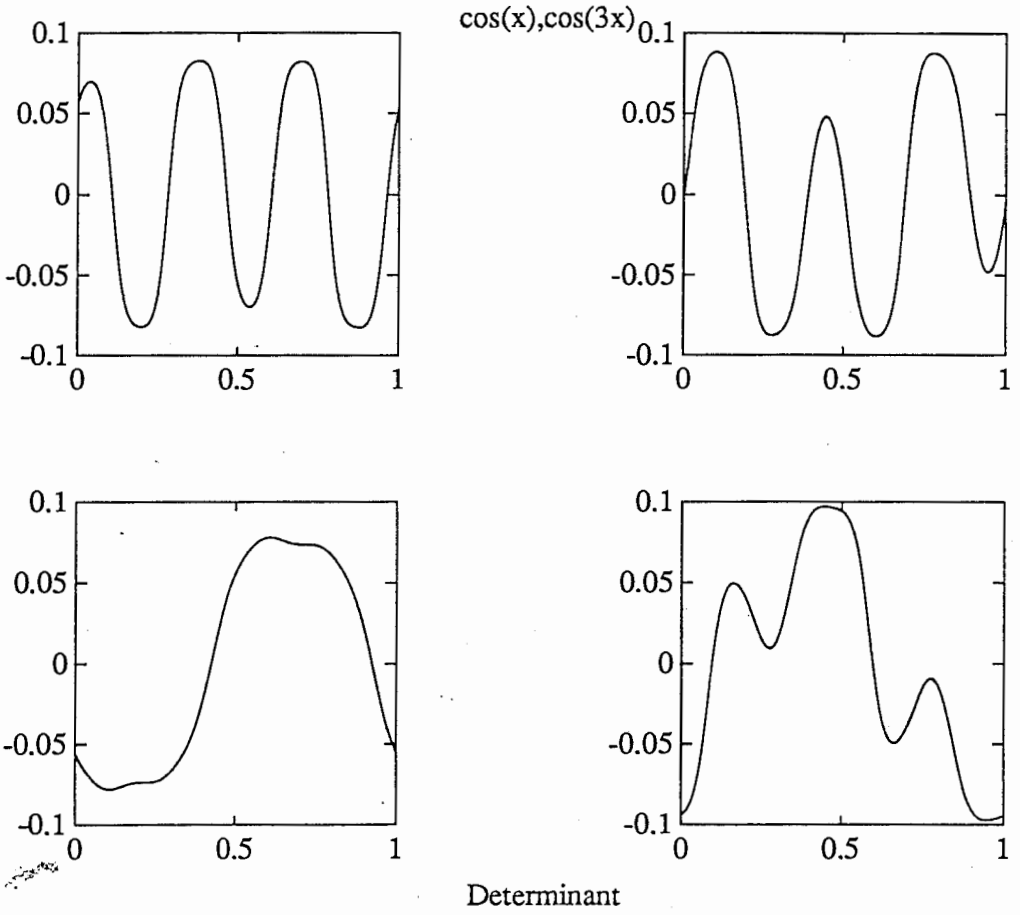

Figure 7: Cos(x), cos(3x) data set, determinant system

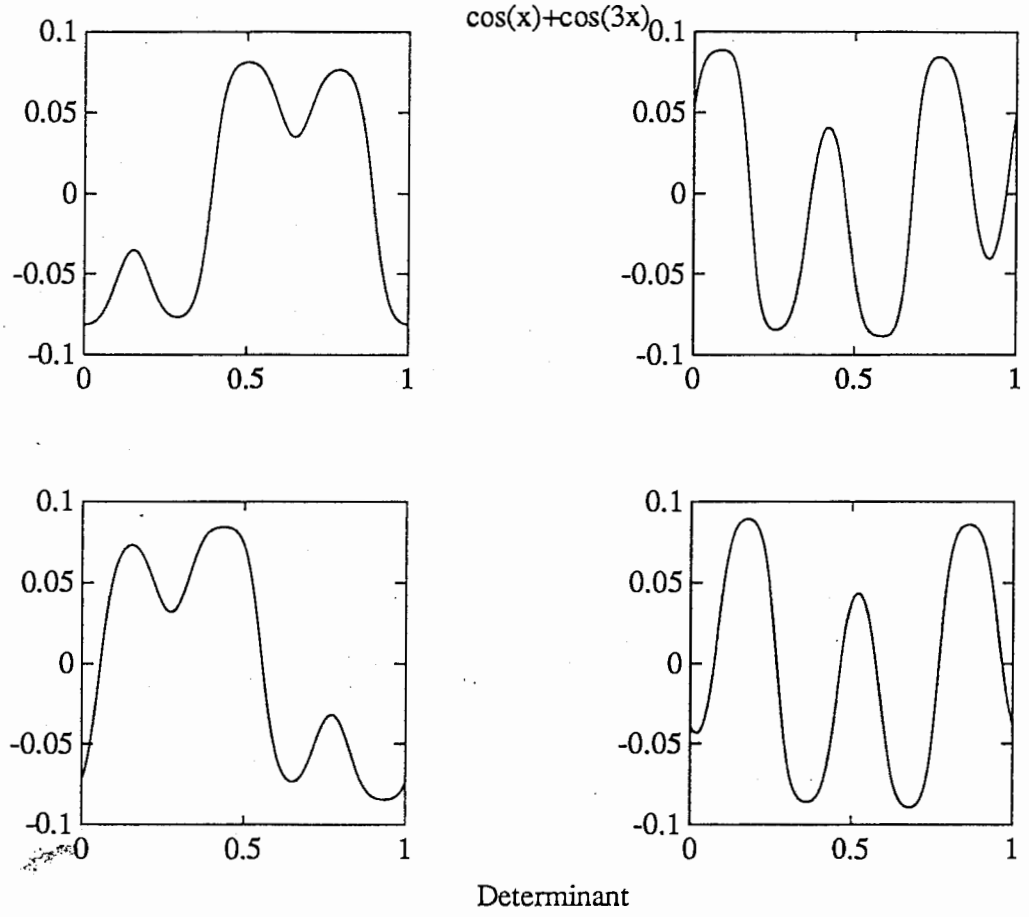

Figure 8: Cos(x)+cos(3x) data set, determinant system

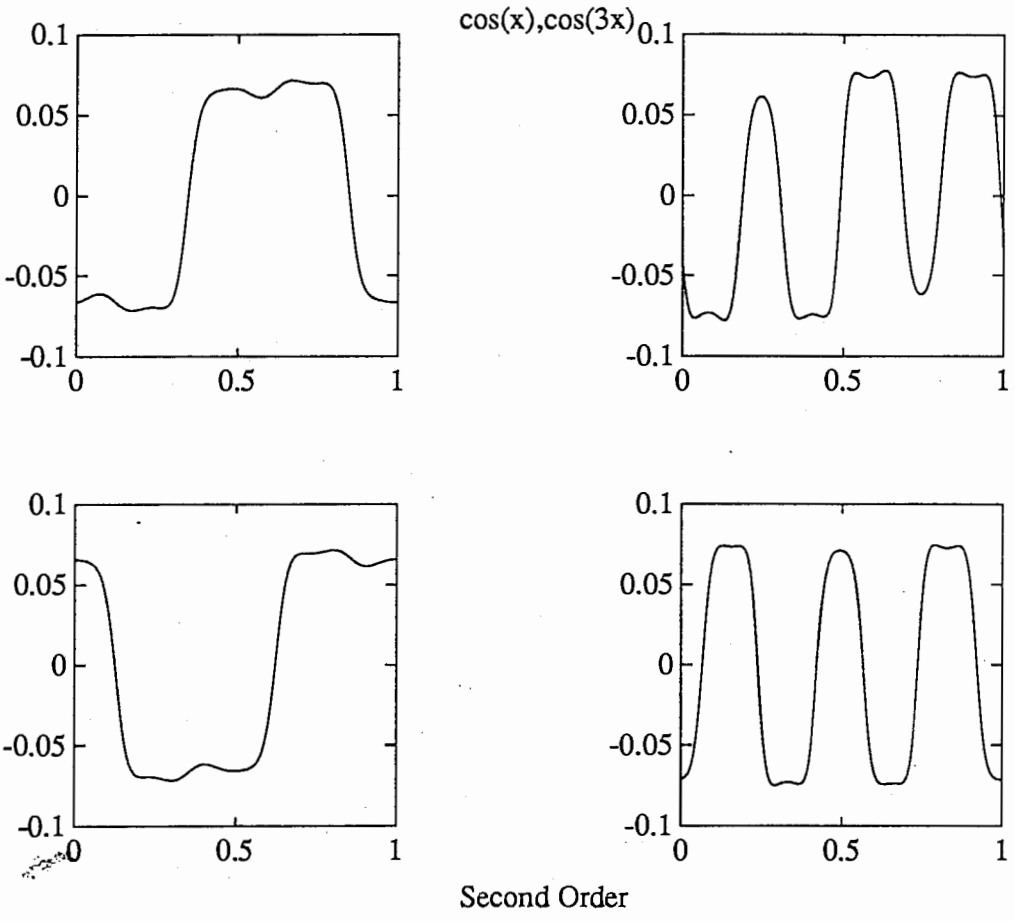

Figure 9: Cos(x), cos(3x) data set, second order system

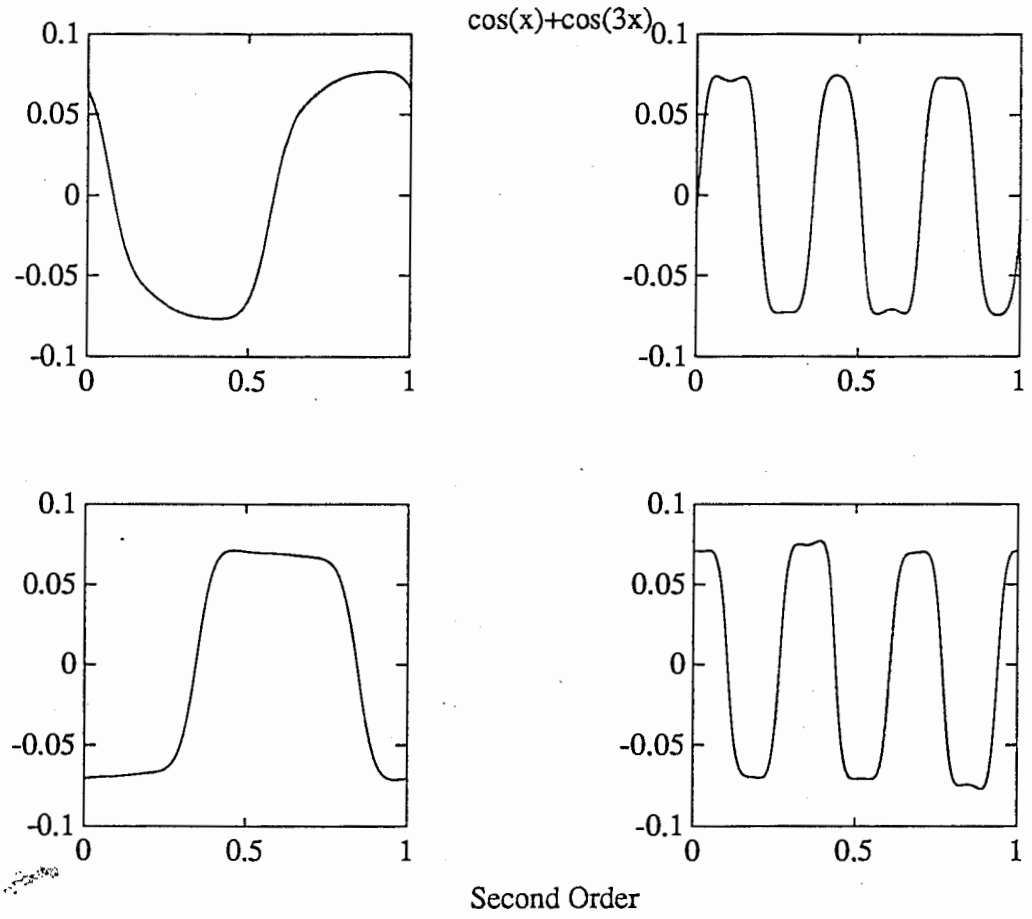

Figure 10: Cos(x)+cos(3x) data set, second order system

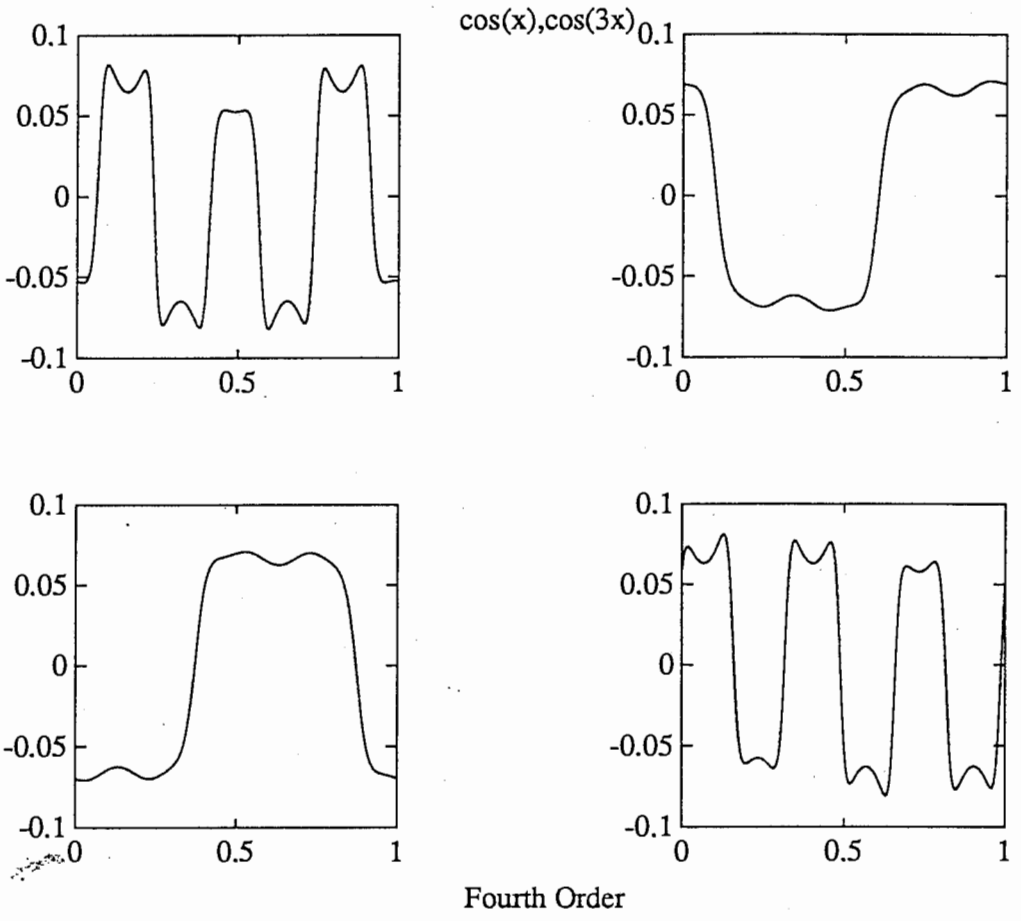

Figure 11: Cos(x), cos(3x) data set, fourth order system

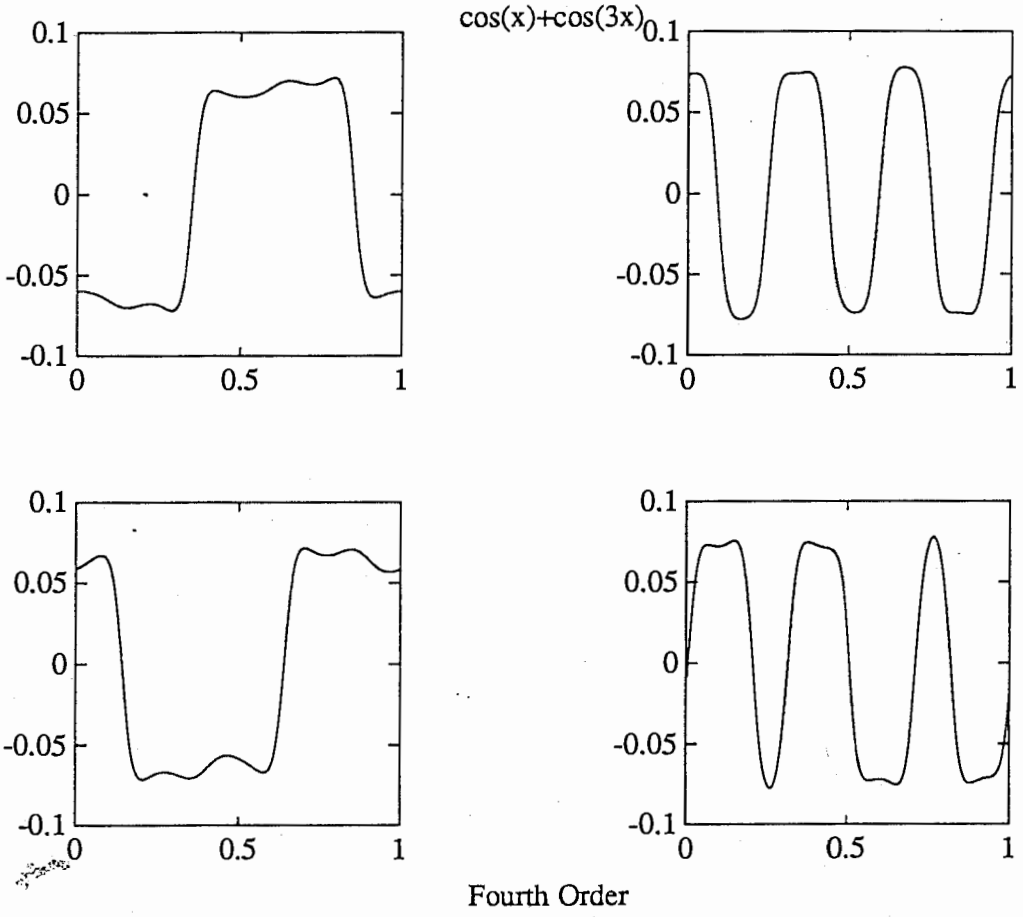

Figure 12: Cos(x)+cos(3x) data set, fourth order system

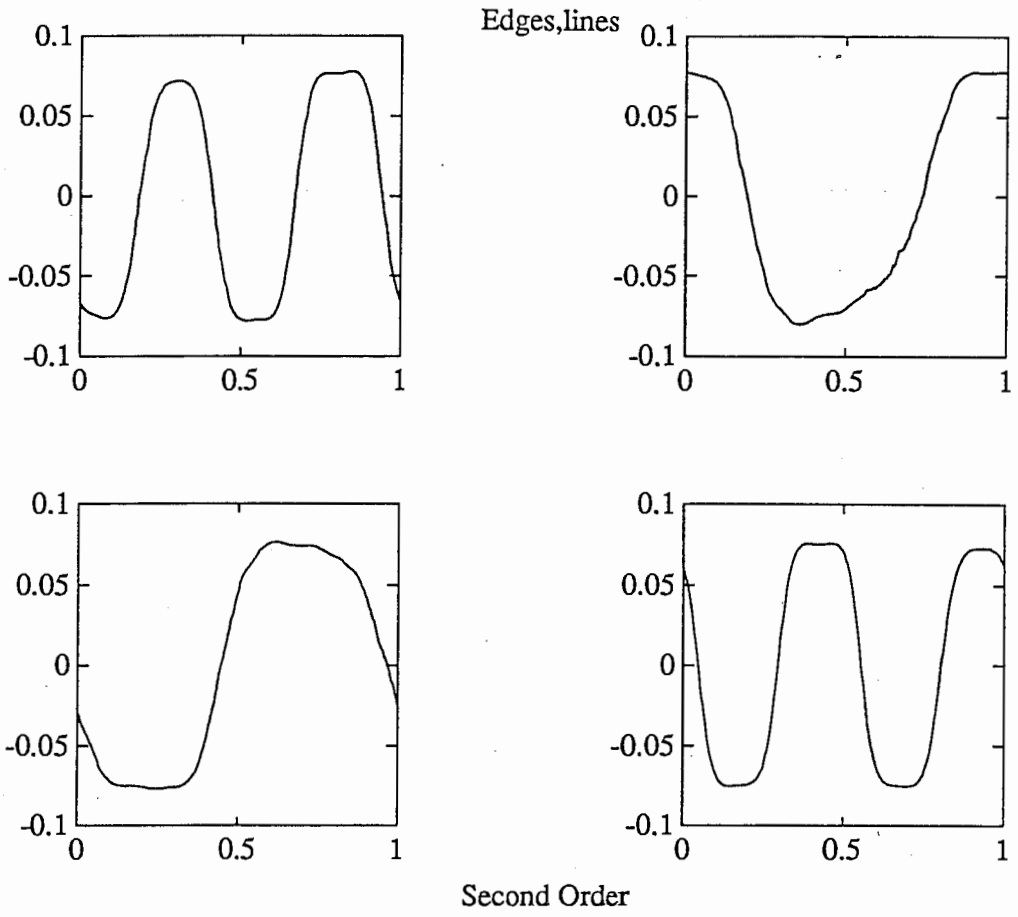

Figure 13: Edge, line data set, second order system

by *edge+line* in the figures) we added one edge and one line pattern first and then we shifted this combined pattern to create new signals. Again we trained the second and the fourth order systems with 2000 examples from these two data sets. The results are shown in the figures 13, 14 for the second order system and figures 15, 16 for the fourth order system.

We see that the second order system stabilizes in the same solution whereas the fourth order system produces two significantly different filter sets.

To give a feeling for the convergence speed we show in figure 17 the values of the quality functions for the determinant, the second order and the fourth order system in the case where we trained them with the patterns cos(x)+cos(3x). We see that the quality of the system does not change significantly after a few hundred iterations.

The last four figures do not only demonstrate that the fourth order system is sensitive to higher order statistics (whereas the second order system is not) but they also show that the systems perform a Fourier analysis of the input process: the filter functions are the sin(x), cos(x), sin(2x), cos(2x) functions. Theoretically this is to be expected since the Fourier transform diagonalizes operators that are translational invariant.

Next we investigate the properties of the filter systems when they are trained with neighborhood data from images. In the figure 2 we saw that the application of the group theoretically constructed coordinate transformation

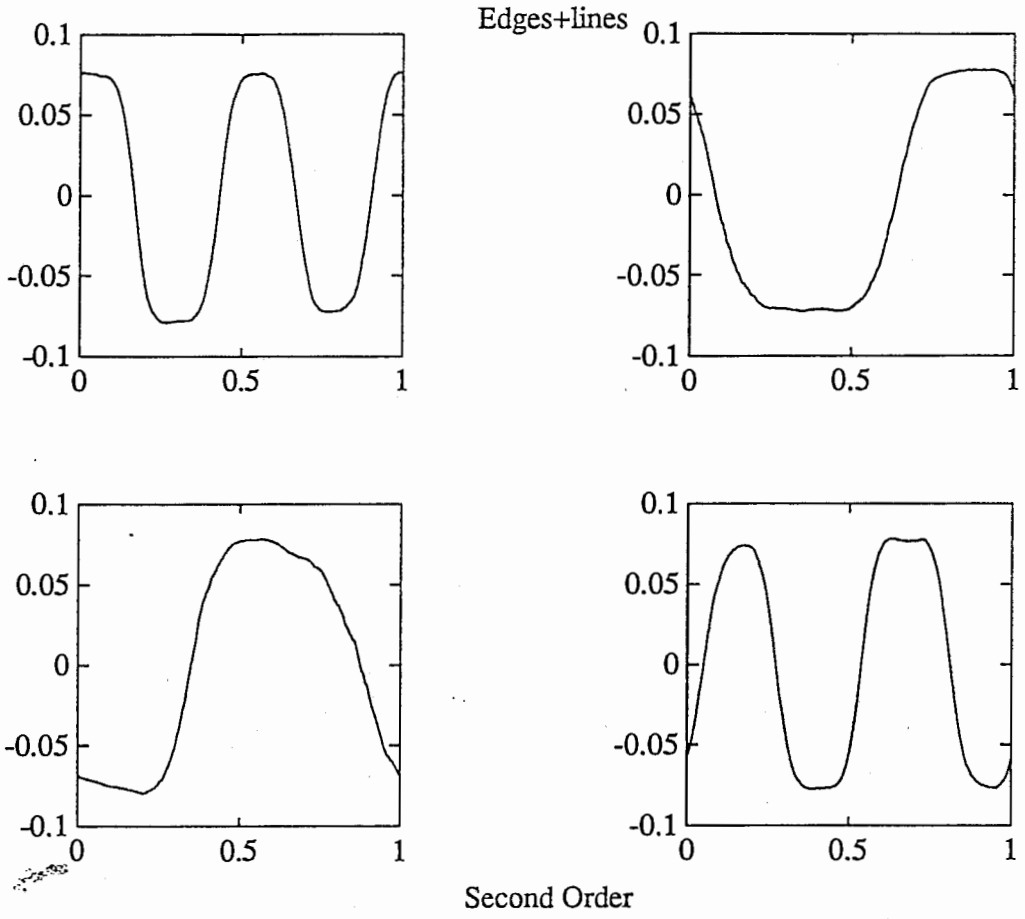

Figure 14: Edge+line data set, second order system

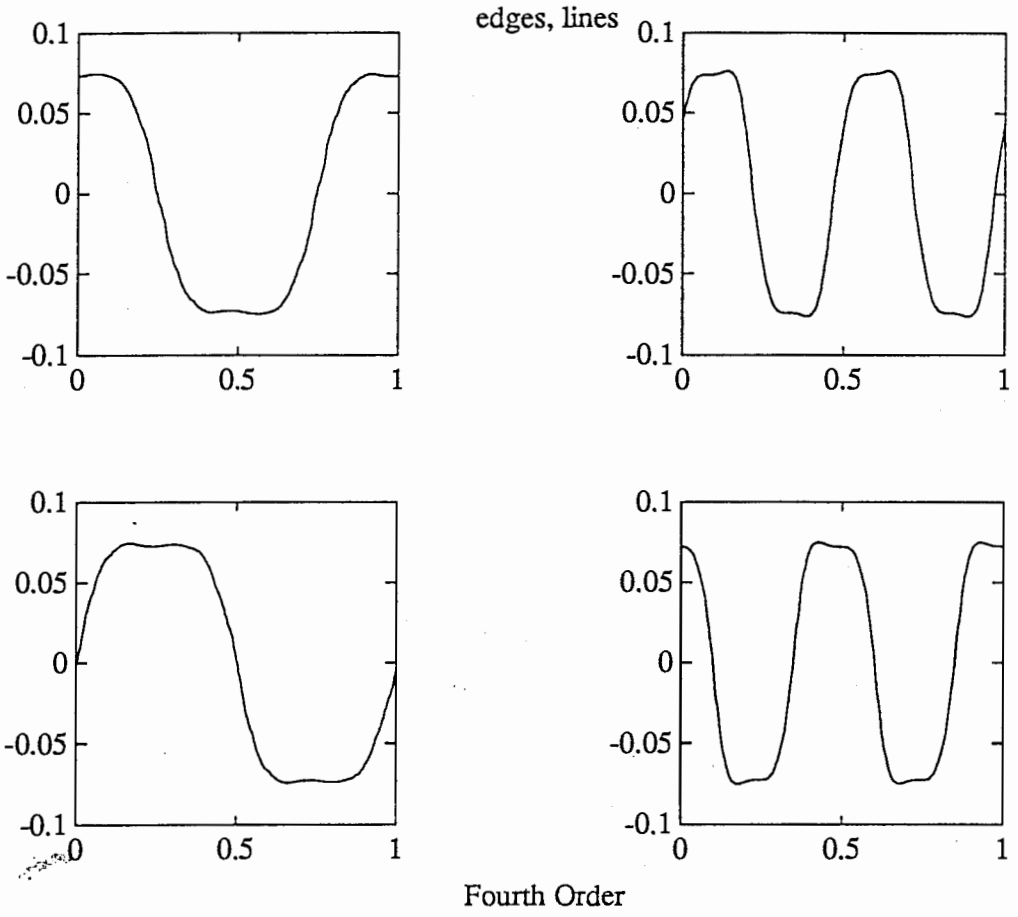

Figure 15: Edge, line data set, fourth order system

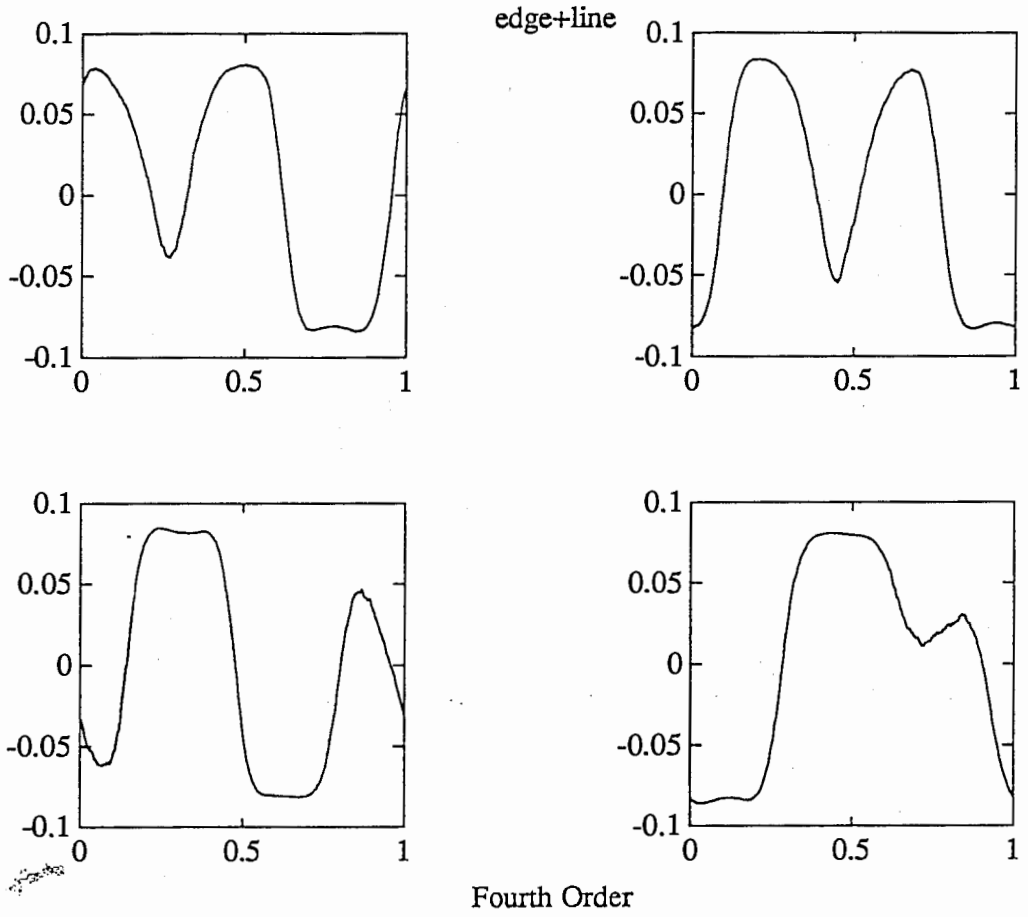

Figure 16: Edge+line data set, fourth order system

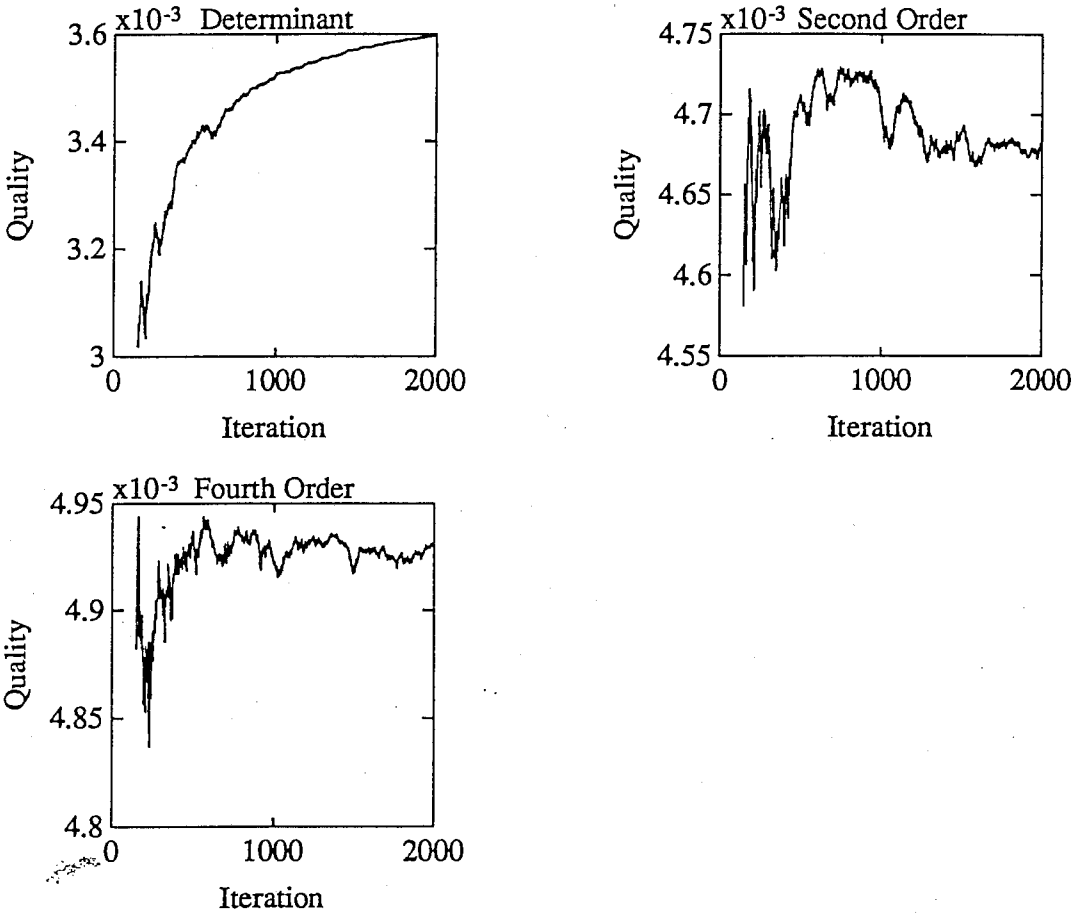

Figure 17: Quality functions for the systems

matrix M lead to a partial decoupling of the feature components. On the other hand we saw in the derivation of the unsupervised filter systems that they too try to decouple the different features by learning the Karhunen-Loeve expansion of the input process. It seems therefore be natural to use the group theoretically derived coordinate transform as a pre-processing process for the unsupervised filter system. The next series of images will describe a few of our experiments based on this idea.

The input image for the next experiments is shown in figure 18.

In the first experiment we learned the four filter functions defined on 5×5 neighborhoods. We trained a fourth order system with 5000 randomly selected neighborhoods from the original image. Since we always assume that the input signals to the system are normed and centered we have to normalize the trainings samples first (we ignore all samples with a variance lower then a small threshold value). Filtering a subimage of the original image with the resulting filters give the results shown in figure 19 Next we applied the group-theoretically defined coordinate transformation as a pre-processing step. Since the transformation matrix is complex we have to concatenate the real part and the imaginary part of the transformed data vector to produce a new data-vector. In our case this gives a 50-dimensional input vector. The system was again trained with 5000 neighborhoods and the resulting filter system was then transformed back to the image coordinate system. Filtering the subimage with these filters resulted in the images shown in figure 20

The sum of the absolute values of the four filtered images represents the accumulative response of the system. The two filter systems give comparable results as can be seen in figure 21 where we show the inverted images obtained from the two systems (the system trained on original image is shown on the top and the transformed system on the bottom).

In the next two plots (figure 22) we see the dynamical behaviour the quality functions for the two systems. In the upper plot (showing all 5000 iterations) we see that the overall behaviour for the two systems is the same. In the lower plot (with only the first 50 iterations) we see that the quality function for the system trained with the transformed input data (dashed line) reacts faster to the incoming data.

In the next series of figures we see the correlation between the learned filter functions and the eigenvectors of the correlation matrix. In this experiment we selected randomly 10000 neighborhoods of size 5×5 from th original image. From this data set we computed the correlation matrix. Then we computed the scalar products between the learned filter functions and the eigenvectors of the estimated correlation matrix. The results are shown in the figures 23 (for the system trained with the original data) and 24 (for the system with the transformed data). We see that both systems have strong correlations with the eigenvectors. In the following tables we give the results of the matrix products $M_o = F_o \cdot F'_o, M_t = F_t \cdot F'_t, N_o = F_o \cdot C \cdot F'_o$ and $N_t = F_t \cdot C \cdot F'_t$ where F_o and F_t are the matrices containing the filter coefficients of the system trained with the original data and the system trained with the transformed data respectively. C is the estimated correlation matrix. These matrices show how correlated the different filters are and how good they approximate the

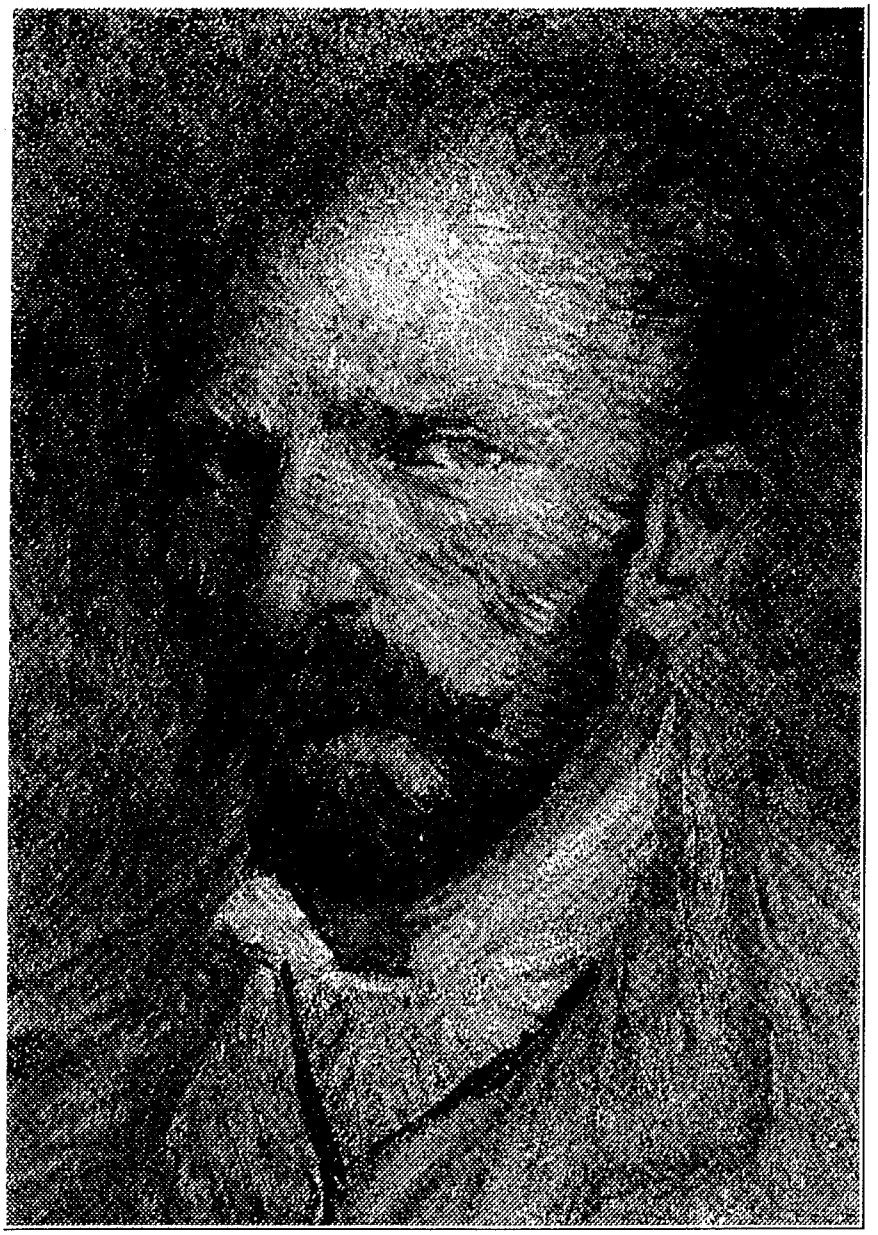

Figure 18: The original image

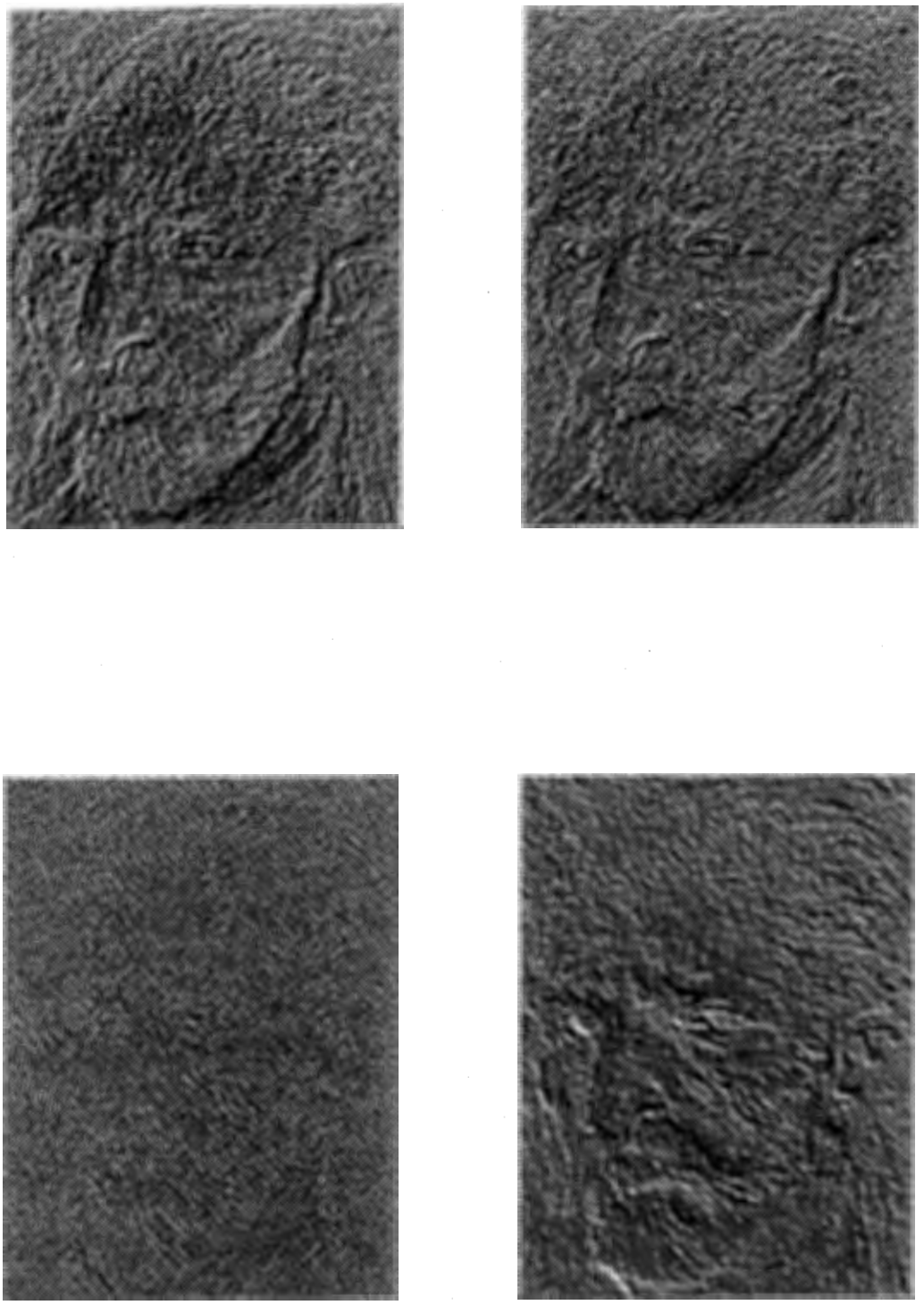

Figure 19: Filtering with the filters learned from the original image

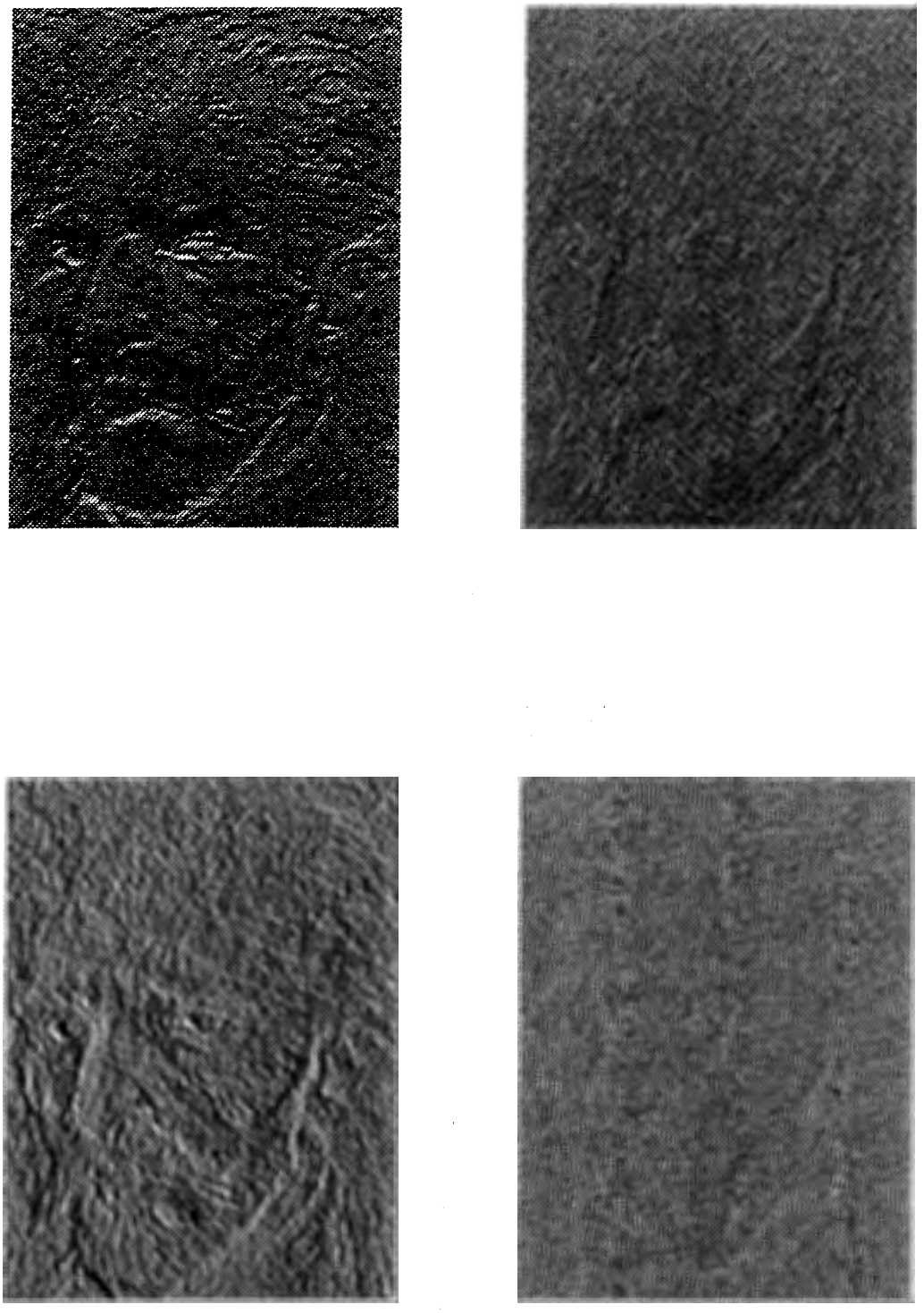

Figure 20: Filtering with the filters learned from the transformed image

Figure 21: Sum of absolute responses (original and transformed)

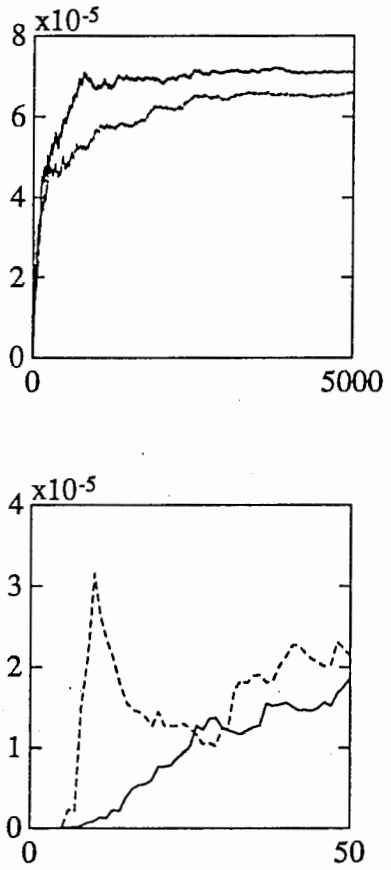

Figure 22: The values of the quality functions

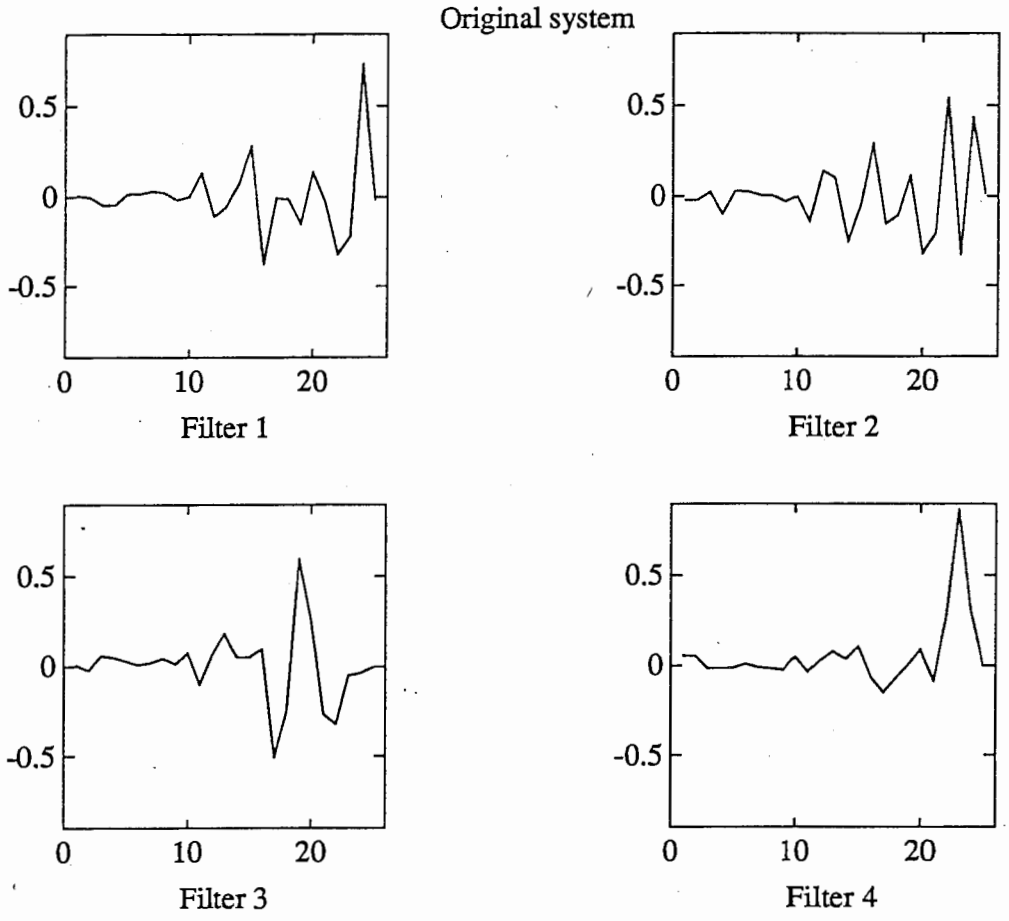

Figure 23: Correlation between the eigenvectors and the learned filters (original)

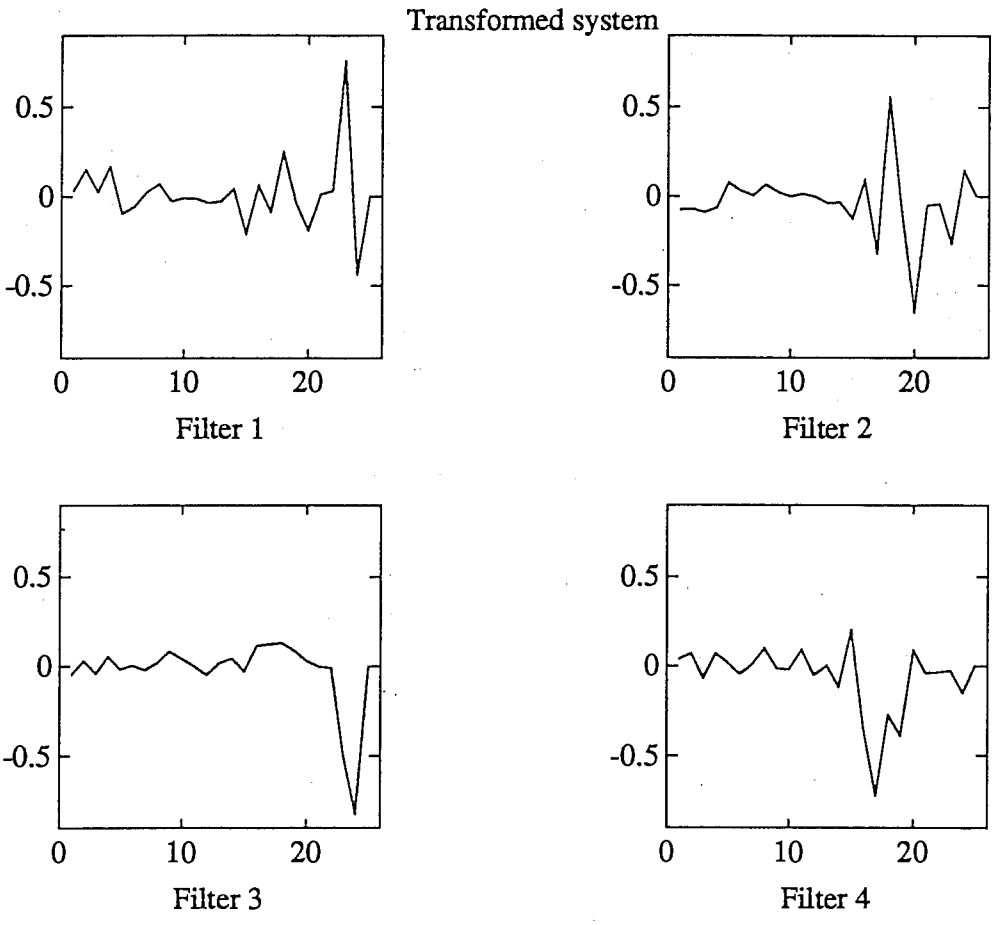

Figure 24: Correlation between the eigenvectors and the learned filters (original)

eigenvectors of the correlation matrix:

$$M_o = \begin{pmatrix} 1.0000 & -0.0155 & -0.0034 & 0.0116 \\ -0.0155 & 1.0000 & 0.0214 & -0.0052 \\ -0.0034 & 0.0214 & 1.0000 & 0.0237 \\ 0.0116 & -0.0052 & 0.0237 & 1.0000 \end{pmatrix} \tag{25}$$

$$M_t = \begin{pmatrix} 1.0000 & 0.0269 & 0.0210 & 0.0010 \\ 0.0269 & 1.0000 & 0.0358 & -0.0110 \\ 0.0210 & 0.0358 & 1.0000 & -0.0558 \\ 0.0010 & -0.0110 & -0.0558 & 1.0000 \end{pmatrix} \tag{26}$$

$$N_o = \begin{pmatrix} 0.6678 & 0.1059 & 0.0115 & 0.0153 \\ 0.1059 & 0.5888 & -0.0208 & -0.0226 \\ 0.0115 & -0.0208 & 0.4656 & -0.0199 \\ 0.0153 & -0.0226 & -0.0199 & 0.7249 \end{pmatrix} \tag{27}$$

$$N_t = \begin{pmatrix} 0.6817 & -0.0672 & 0.0199 & 0.0154 \\ -0.0672 & 0.4841 & 0.0120 & -0.0123 \\ 0.0199 & 0.0120 & 0.7663 & 0.0254 \\ 0.0154 & -0.0123 & 0.0254 & 0.4446 \end{pmatrix} \tag{28}$$

We see that the filters in transformed system are slightly more correlated and that the two strongest filter functions have slightly higher values in the diagonals. The overall response (measured by the value of the traces of N_o and N_t) is comparable.

Finally we tried to restore the original image by adding the mean value in the neighborhood and the computed filter results. The results are shown in figures 25 and figures 26 for the system trained with the original data and the system trained with the transformed data.

In this case we found that the system trained with original data and the system trained with transformed data produced very simular results. In the next two series of experiments we will show that the system trained with the transformed neighborhoods may provide visually better results.

In the first experiment we trained a system consisting of four units with 20000 samples from an image. The neighborhood size was again 5×5 pixels. Then we filtered a subimage with the four filter functions. The sum of the absolute values of the four response images is shown in figure 27 (the system with the original data on the top and the one with the transformed data on the bottom). Then we smoothed the subimage with a 5×5 averaging filter and added the smoothed image and the four filter response images. The result is shown in figure 28. The image on the bottom is the original subimage, the top left image was produced by the system with the original data and the top right image comes from the transformed data. We see that the system trained with the original data produces a more blurred image than the system that works in the transform domain. This observation is confirmed when we trained a four unit system with the Baboon image. In this experiment we used only 2500 neighborhoods from the original image. The same results (absolute response and restorations) are shown in the images 29 and 30: In this case some of the filters learned in the transform domain had also a simple interpretation: they were two perpendicular edge filters. The four filters learned by the system trained in the transform domain are shown in figure 31 and the four filters extracted from the original data are shown in figure 32.

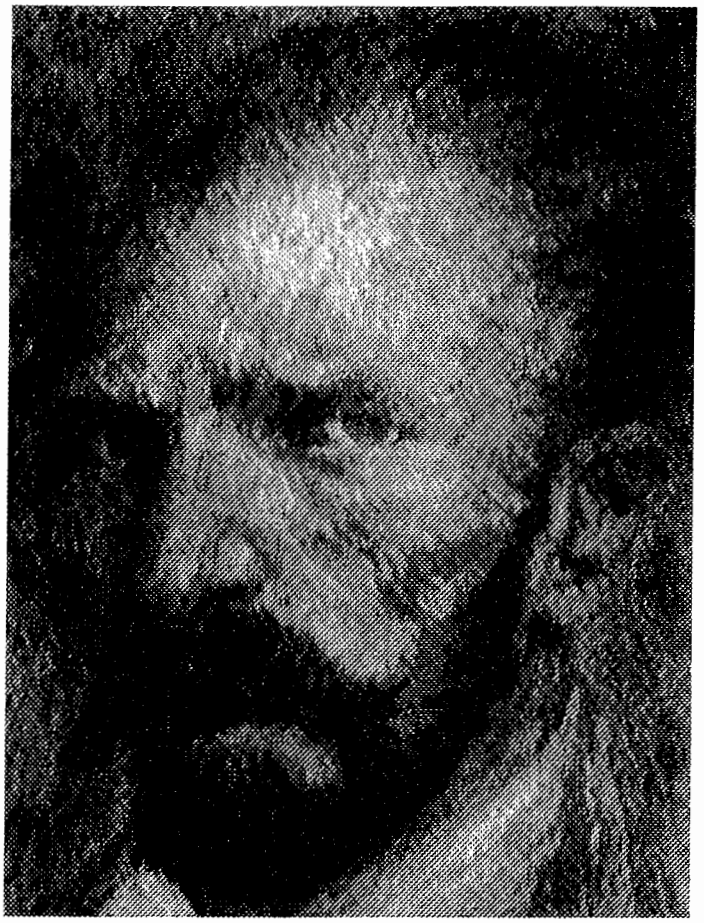

Figure 25: Restoration with mean and filter results (original data)

Figure 26: Restoration with mean and filter results (transformed data)

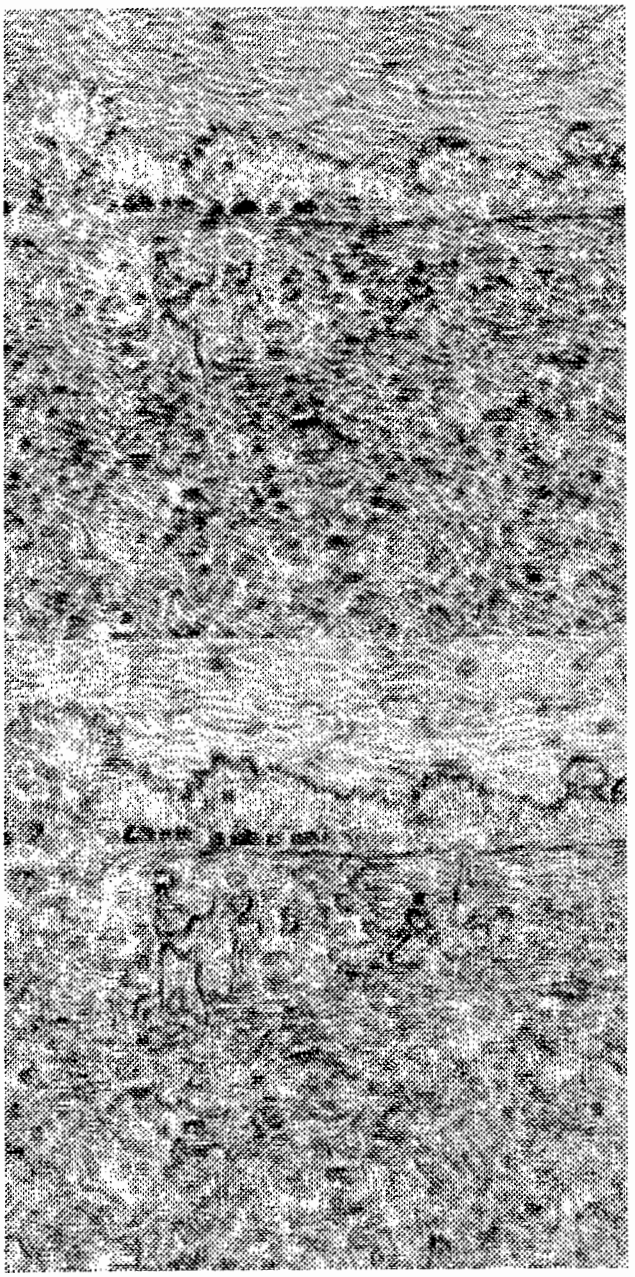

Figure 27: Inverted absolute reponse of the systems (original top; transformed bottom)

Figure 28: Restoration with mean and filter results (restoration with original, with transform; original image)

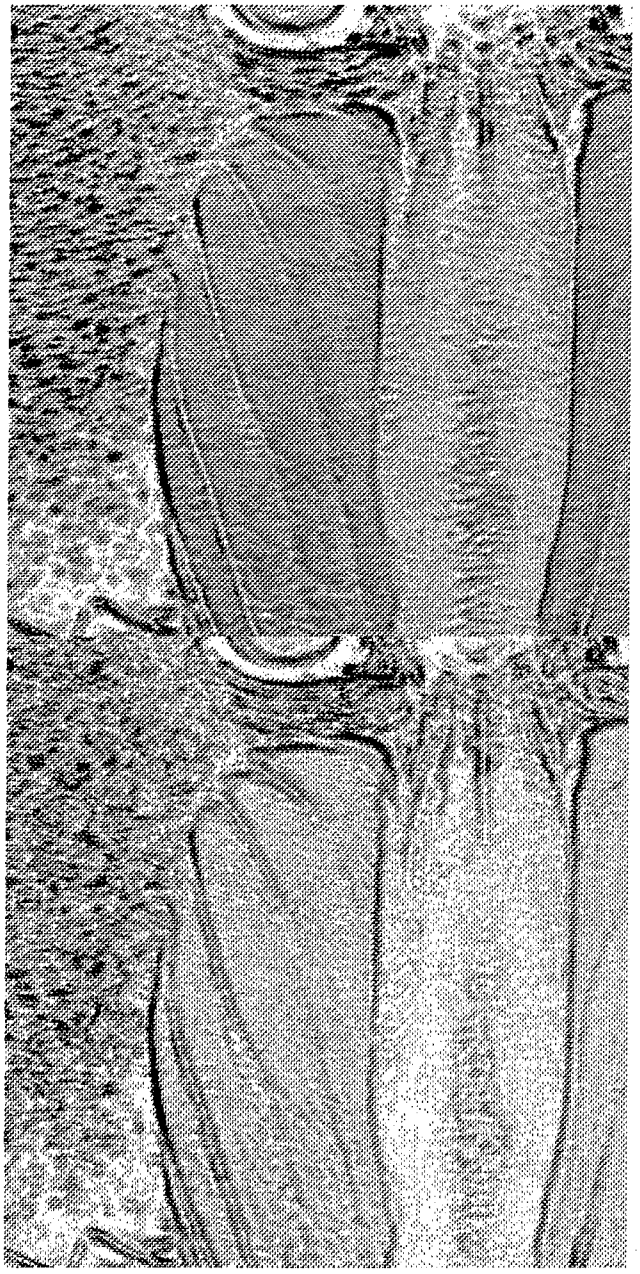

Figure 29: Inverted absolute reponse of the systems (original top; transformed bottom)

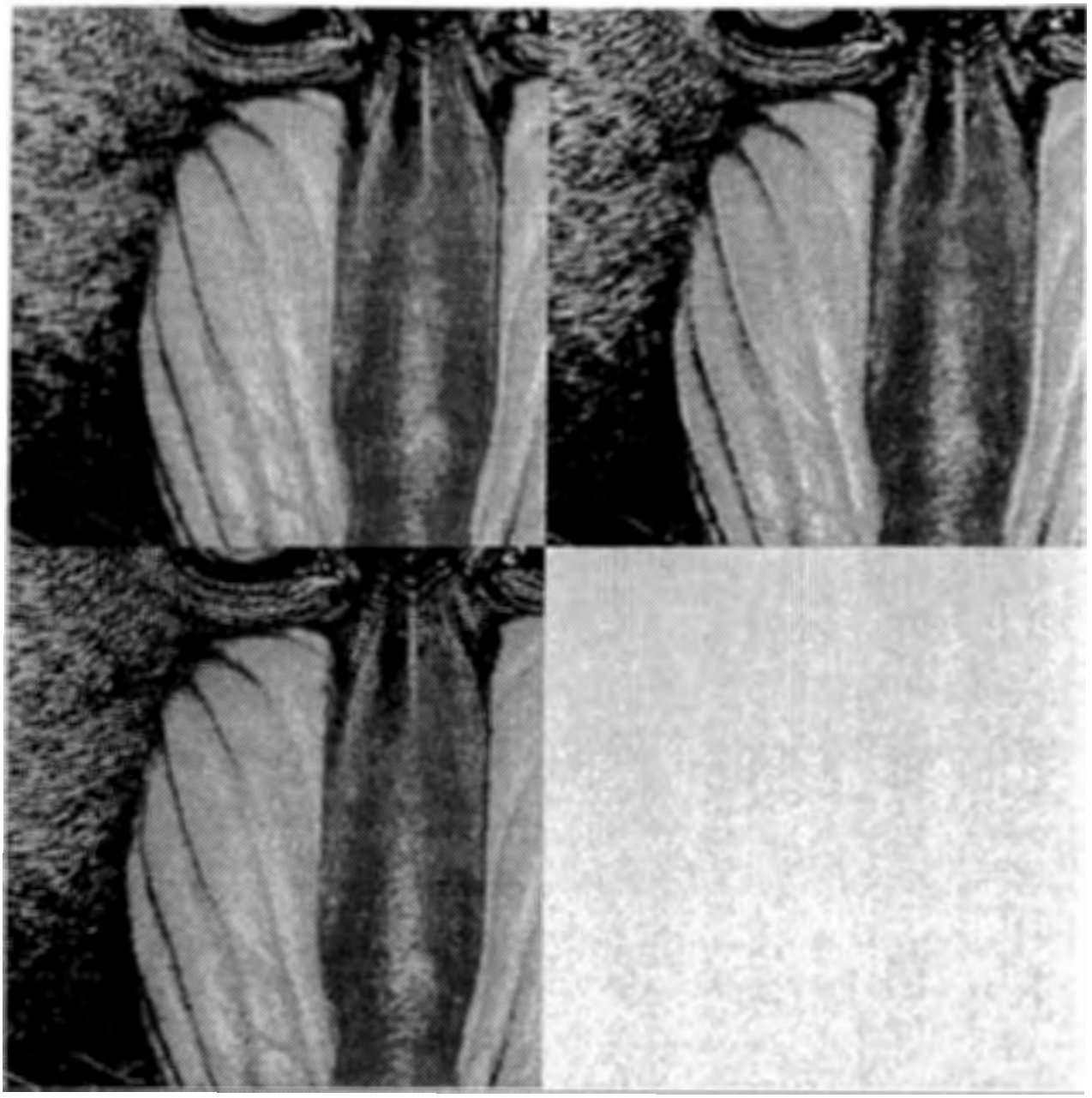

Figure 30: Restoration with mean and filter results (restoration with original, with transform; original image)

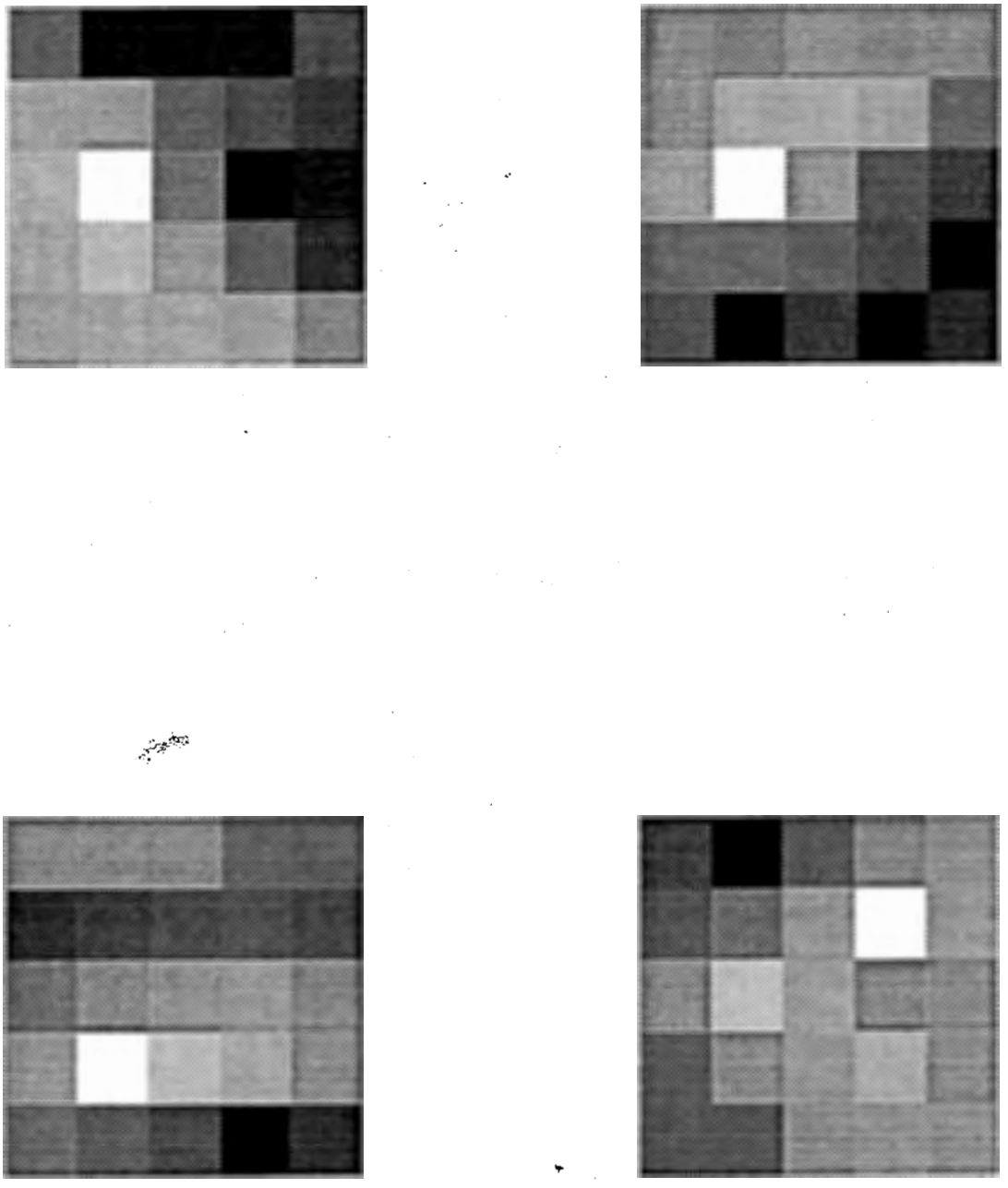

Figure 31: Four filter kernels learned by the system trained in the transform domain

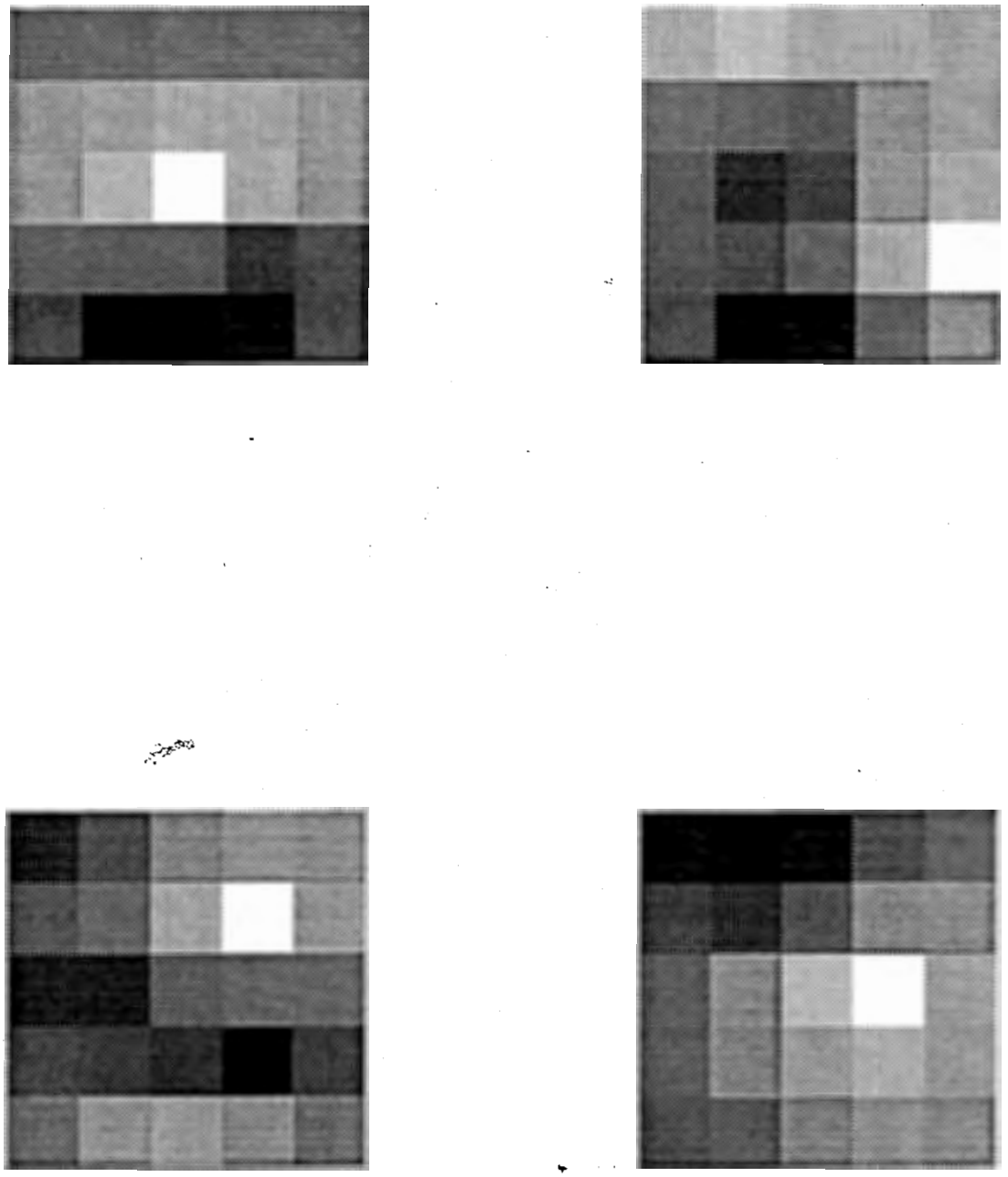

Figure 32: Four filter kernels learned by the system trained in the original domain

From these experiments we conclude that it might be preferable to use the group-theoretically motivated coordinate transform as a pre-processing stage for a learning filter system. This might increase the speed of learning and the results produced with these systems seem to be superior. The additional computations can be done very fast since they involve only additions and subtractions.

7 Conclusions

We demonstrated first how problems on a regular grid can be simplified by using results from the representation theory of the dihedral group. The main result was that a whole class of problems can be simplified by applying a group theoretically derived coordinate transformation. Then we introduced several unsupervised filter systems that can learn the Karhunen-Loeve expansion in parallel. We also discussed why Karhunen-Loeve type expansions might not be sufficient in many cases. This lead us to the introduction of a learning filter system based on fourth order moments. We investigated the properties of this system with a number of generated data sets. Finally we combined the group theoretically defined coordinate transformation and the learning filter systems to learn filter functions from real images. We showed in some experiments that the systems operating in the transform domain have superior performance compared to those working on the original pixel data.

8 Acknowledgement

The support of K. Shimohara, N. Sonehara, E. Yodogawa and the members of the Cognition group is gratefully acknowledged. I also want the thank Peter Meer for helpful comments on a first draft of the paper.

References

[1] Joseph J. Atick. Could information theory provide an ecological theory of sensory processing? *Network*, 3(2):213–251, May 1992.

[2] Hermann Boerner. *Darstellungen von Gruppen.* Springer Verlag, Berlin, Göttingen, Heidelberg, 1955.

[3] J. Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(6), 1986.

[4] C. W. Curtis and I. Reiner. *Representation Theory of Finite Groups and Associative Algebras.* Wiley, New York, 1962.

[5] E. Hildreth D. Marr. Theory of edge detection. *Proceedings Royal Society London*, B-207:187–217, 1980.

[6] Per-Erik Danielsson. Rotation invariant linear operators with directional response. In *Proc. 5. ICPR*, pages 1171–1176, 1980.

[7] I. M. Gelfand, R. A. Minlos, and Z. Y. Shapiro. *Representations of the rotation and Lorentz groups and their applications.* Pergamon Press, 1963.

[8] Manfred H. Hueckel. An operator which locates edges in digitized pictures. *Journal of the Association for Computing Machinery*, 18(1):113–125, 1971.

[9] Robert A. Hummel. Feature detection using basis functions. *Computer Graphics and Image Processing*, 9:40–55, 1979.

[10] Ken ichi Kanatani. *Group Theoretical Methods in Image Understanding.* Springer Verlag, 1990.

[11] J. J. Koenderinck and Andrea J. van Doorn. Generic neighborhood operators. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(6):597–605, June 1992.

[12] J. J. Koenderinck and Andrea J. van Doorn. Receptive field assembly pattern specificity. *Journal of Visual Communication and Image Representation*, 3(1):1–12, 1992.

[13] J.J. Koenderinck and A. J. van Doorn. Receptive field families. *Biological Cybernetics*, 63:291–297, 1990.

[14] T. K. Leen. Dynamics of learning in linear feature-discovery networks. *Network*, 2(1):85–105, 1990.

[15] Reiner Lenz. Optimal filters for the detection of linear patterns in 2-d and higher dimensional images. *Pattern Recognition*, 20(2):163–172, 1987.

[16] Reiner Lenz. A group theoretical model of feature extraction. *Journal of the Optical Society of America A*, 6(6):827–834, 1989.

[17] Reiner Lenz. *Group Theoretical Methods in Image Processing.* Lecture Notes in Computer Science (Vol. 413). Springer Verlag, Heidelberg, Berlin, New York, 1990.

[18] Reiner Lenz. On probabilistic invariance. *Neural Networks*, 4(5), 1991.

[19] Reiner Lenz and Mats Österberg. Learning filter systems. In *Proc. Int. Neural Networks Conference, Paris*, 1990.

[20] Reiner Lenz and Mats Österberg. Learning filter systems. In *Proc. Int. Joint Conference on Neural Networks, San Diego*, 1990.

[21] Reiner Lenz and Mats Österberg. A parallel learning filter system that learns the kl-expansion from examples. In *Proc. First IEEE-SP Workshop on Neural Networks for Signal Processing (Sept. 91)*, 1991.

[22] Reiner Lenz and Mats Österberg. A new method for unsupervised linear feature extraction using fourth order moments. *Pattern Recognition Letters*, Accepted.

[23] Reiner Lenz and Mats Österberg. Computing the karhunen-loeve expansion with a parallel, unsupervised filter system. *Neural Computations*, In print.

[24] Peter Meer and Isaac Weiss. Smoothed differentiation filters for images. *Journal of Visual Communication and Image Representation*, 3(1):58–72, March 1992.

[25] M. A. Naimark and A. I. Stern. *Theory of Group Representations.* Springer Verlag, New York, Heidelberg, Berlin, 1982.

[26] E. Oja. A simplified neuron model as a principle component analyser. *Journal of Mathematical Biology*, 15:267–273, 1982.

[27] E. Oja. Neural networks, principal components, and subspaces. *Int. Journal of Neural Systems*, 1:61–68, 1989.

[28] Mats Österberg and Reiner Lenz. Learning filter systems: Basic principles and some applications. In *7th Scandinavian Conference on Image Analysis*, 1991.

[29] Pietro Perona. Steerable-scalable kernels for edge detection and junction analysis. In *Proc. European Conference Computer Vision-92*, pages 3–23. Springer, 1992.

[30] Terence D. Sanger. Optimal unsupervised learning in a single-layer linear feedforward neural network. *Neural Networks*, 2(6):459–474, 1989.

[31] Bart M. ter Haar Romeny, Luc M. J. Florack, Jan J. Koenderink, and Max A. Viergever. Scale-space: its natural operators and differential invariants. In *Proc. Information Processing in Medical Imaging, IPMI'91*, volume 511 of *Lecture Notes in Computer Science*, pages 239–255. Springer, 1991.

A Basic facts from the theory of group representations

In this appendix we will summarize some of the very basic definitions and results from the representation theory of finite groups with special emphasis to the dihedral group. It contains only the most important facts needed in this paper. To get more information the reader might consult the literature (see [17], [10], [4], [2], and [25]).

A.1 Groups

We recall that a **group** is a set G together with a group operation $\cdot$ such that:

1. $\cdot : G \times G \to G; (g_1, g_2) \mapsto g_1 \cdot g_2$
2. There is a **neutral element** e such that

$$e \cdot g = g \cdot e = g \text{ for all } g \in G \tag{29}$$

3. For each $g \in G$ there is an **inverse element** g^{-1} such that

$$g^{-1} \cdot g = g \cdot g^{-1} = e \tag{30}$$

4. The operation is **associative**

$$g_1 \cdot (g_2 \cdot g_3) = (g_1 \cdot g_2) \cdot g_3 = g_1 \cdot g_2 \cdot g_3 \text{ for all } g_1, g_2, g_3 \in G. \tag{31}$$

Usually we will write $g_1 g_2$ instead of $g_1 \cdot g_2$.

Definition 1
1. The group is called **abelian** or **commutative** if $g_1 g_2 = g_2 g_1$ for all $g_i \in G$.
2. A group is called **cyclic** if there is an element $g_0 \in G$ such that all group elements g have the form $g = g_0^n$ for some integer n.
3. A group is called a **finite** group if it consists of finitely many elements.
4. The number of elements in a group is called the **order** of the group. We will usually denote it by $|G|$.

Sometimes it is enough to know a few elements of the group since all the other elements can be generated from them. This is especially important for the dihedral groups:

Definition 2 Assume G is a group and $S = \{g_i : i \in I\}$ is a subset of G. Then we call S a **generating system of** G if all elements $g \in G$ can be written as finite products of elements in S :

$$g = g_1 ... g_N \quad (g_n \in S) \tag{32}$$

The dihedral group $\mathcal{D}_n$ is defined as the group of all isometries that leave a regular, n-sided polygon invariant. Using some geometry it can be shown that the Dihedral group $\mathcal{D}_n$ consists of 2n elements: n rotations and n transpositions. The rotations belonging to $\mathcal{D}_n$ form a cyclic group $\mathcal{C}_n$ of order n. This group consists of all 2-D rotations with rotation angles $\frac{2k\pi}{n}, (k = 0, ..., n-1)$.

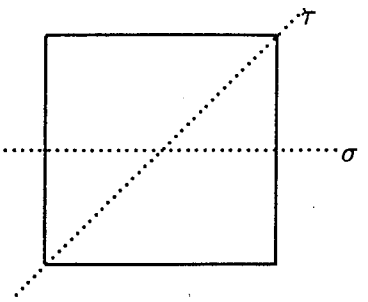

Figure 33: Generating elements for the group $\mathcal{D}_4$

The other n elements in the group are the transpositions on the lines with angle $\frac{k\pi}{n}, (k = 0, ..., n-1)$ with the x-axes (we always assume that the first axis of the polygon lies on the x-axis).

For the Dihedral group $\mathcal{D}_4$ (connected to the square) two of these transpositions are denoted by σ and τ in the figure 33. The 90° rotation will be denoted by ρ.

From the figure we derive the following properties of $\mathcal{D}_4$:

Theorem 2

1. The elements σ and τ form a generating system of $\mathcal{D}_4$.
2. Applying two transpositions gives a rotation: $\tau\sigma = \rho$ and $\sigma\tau = \rho^{-1}$
3. The elements σ and ρ form a generating system of $\mathcal{D}_4$
4. $\sigma\rho^k = \rho^{-k}\sigma$
5. The elements of $\mathcal{D}_4$ are all of the form: $\sigma^l\rho^k$ with $l = 0, 1$ and $k = 0, 1, 2, 3$.

Similar relations hold for all dihedral groups.

A.2 Representations

We now come to the important definition of a (finite-dimensional) representation:

Definition 3 Assume G is a group, V is a vector space and $GL(V)$ is the space of all invertible, linear maps: $V \to V$. A map $T : G \to GL(V)$ is called a **representation** of G if

$$T(g_1g_2) = T(g_1)T(g_2) \text{ for all } g_1, g_2 \in G \tag{33}$$

The dimension of V is called the **dimension of the representation.** If all the mappings $T(g), g \in G$ are unitary then we say that the representation is **unitary.**

In the following we will only consider finite-dimensional representations. If we select one fixed basis in the vector space then we can think of $T(g)$ as a matrix.

If S is a generating system of the group G then it is sufficient to define the representation mapping T on the elements of the generating system S. For the dihedral group $\mathcal{D}_4$ we have $T(g) = T(\sigma^l\rho^k) = T(\sigma)^l T(\rho)^k$ with $l = 0, 1$

and $k = 0, 1, 2, 3$. The representation is therefore completely defined by the two matrices $T(\sigma)$ and $T(\rho)$.

Next we define the simplest representations possible:

Definition 4

1. Assume $T : G \to L(V)$ is a representation of G. Assume that V_1 is a subspace of V that is also invariant under all operations $T(g), g \in G$. Then we call V_1 an **invariant subspace.**
2. A representation T is called **reducible** if there is an invariant subspace V_1 of V such that $V_1 \neq \{ 0 \}$ and $V_1 \neq V$.
3. A representation T is called **irreducible** if it is not reducible.

An irreducible representation is thus a representation with no non-trivial invariant subspaces. The invariant subspaces of an irreducible representation are the smallest possible.

In terms of matrices we get the following characterizations:

Theorem 3 The representation $\{T(g_0), ..., T(g_{N-1})\}$ of a finite group is reducible if and only if there is an invertible matrix P such that all matrices $PT(g_k)P^{-1}$ have the form:

$$PT(g_k)P^{-1} = \begin{pmatrix} A_k & B_k \\ 0 & C_k \end{pmatrix} \tag{34}$$

where the matrices A_k, B_k and C_k all have the same size for all indices k.

If we can further simplify the matrices so that they become block diagonal then we say that the representation is completely reducible:

Definition 5 The representation $\{T(g_0), ..., T(g_{N-1})\}$ of a finite group is **completely reducible** if and only if there is an invertible matrix P such that the matrices $PT(g_k)P^{-1}$ have the form:

$$PT(g_k)P^{-1} = \begin{pmatrix} A_k^0 & 0 & ... & 0 \\ 0 & A_k^1 & ... & 0 \\ . & . & ... & . \\ 0 & 0 & ... & A_k^L \end{pmatrix} \tag{35}$$

where the matrices $\left\{A_0^l, ..., A_{N-1}^l\right\}$ all have the same size for all indices l and where

$$T_l : G \to \left\{A_0^l, ..., A_{N-1}^l\right\}$$

defines an irreducible representation of G for all l.

In the case where the group is finite it can be shown that all representations are essentially unitary and completely reducible. This is an important result since it allows us to simplify calculations considerably.

Assume that G is a finite group and $T : G \to GL(V)$ is a finite-dimensional representation. Then we find:

Theorem 4

1. There is a scalar product in V such that the representation is unitary.
2. All representations of finite groups are completely reducible.

A.3 Representations of the dihedral groups

We recall that a dihedral group is generated by two elements, a rotation ρ and a transposition σ. It is therefore sufficient to define the representation for these two group elements. For the dihedral groups we find the following types of irreducible representations:

1. $T(\rho) = \begin{pmatrix} \zeta_m & 0 \\ 0 & \zeta_{-m} \end{pmatrix}$ and $T(\sigma) = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$.
2. $T(\rho) = 1, T(\sigma) = 1$
3. $T(\rho) = 1, T(\sigma) = -1$
4. $T(\rho) = -1, T(\sigma) = 1$
5. $T(\rho) = -1, T(\sigma) = -1$

where ζ_m is the root of unity $\zeta_m = e^{\frac{2\pi i m}{n}}$.

The cases $T(\rho) = \begin{pmatrix} -1 & 0 \\ 0 & -1 \end{pmatrix}$ and $T(\rho) = -1$ can only occur if n is even since only then we get -1 as a root of unity. This completes our study of the irreducible representations of the dihedral group $\mathcal{D}_n$.

A.4 G-symmetric operators and Schur's Lemma

The intertwining or G-symmetric operators where defined as matrices that commuted with all the representation matrices:

Definition 6 Assume that T is a finite-dimensional representation of a finite group G. We say that C is a **(G,T)-symmetrical operator** if the following equations hold for all $g \in G$:

$$T(g)C = CT(g). \tag{36}$$

If it is clear what representation T we mean then we speak of **G-symmetrical operators.** Sometimes we also say that the matrix has the **symmetry of the group** G or that the matrix is an **intertwining operator.**

These equations restrict the form C can have and a precise characterization of all these C is given in the following theorem known as **Schur's Lemma:**

Theorem 5 Let T_1 and T_2 be two finite-dimensional irreducible representations of the finite group G. Furthermore let C be a G-symmetrical operator. Then we have:

1. $C = 0$ or
2. C is invertible and T_1 and T_2 are equivalent.

With the help of Schur's Lemma one can characterize the G-symmetrical matrices as follows:

Theorem 6 Assume that C is a (G,T)-symmetrical matrix. Assume further that the representation matrices $T(g)$ have the form:

$$T(g) = \begin{pmatrix} \tau_1(g) & 0 & \dots & 0 \\ 0 & \tau_2(g) & \dots & 0 \\ . & . & \dots & . \\ 0 & 0 & \dots & \tau_N(g) \end{pmatrix} \tag{37}$$

where all the matrices $\tau_n(g)$ in turn are block diagonal matrices of the form:

$$\tau_j(g) = \begin{pmatrix} t_j(g) & 0 & ... & 0 \\ 0 & t_j(g) & ... & 0 \\ . & . & ... & . \\ 0 & 0 & ... & t_j(g) \end{pmatrix} \tag{38}$$

and the $t_j(g)$ belong to the j-th irreducible representation of the group. The matrices $t_j(g)$ have the size $n_j \times n_j$ and in $\tau_j(g)$ there are c_j identical copies of $t_j(g)$. N is the number of irreducible representations of the group.

In the coordinate system connected to this form of the representation T the matrix C is a block diagonal matrix and in its diagonal it has n_1 matrices of size c_1, n_2 ..., and n_N matrices of size c_N.

A.5 The projection formulas

The last theorem tells us that all G-symmetrical matrices C have block diagonal form in the a special coordinate system in which $T(g)$ has the form described in equations 37 and 38. What is left is the problem to find for a given representation T a coordinate system in which it has the form 37 and 38. This is solved in the following theorem:

Theorem 7 Assume that we have a complete set of irreducible representations of the group G. We denote them by $D_n(g), n = 1, ..., N$. Assume further that the dimension of the representation D_j is n_j. The matrices $D_j(g)$ have the size $n_j \times n_j$ and the element with index (k, l) in the matrix $D_j(g)$ is denoted by $d_{kl}^{(j)}(g)$.

For each triplet (k, l, j) we define now the new matrix $P_{kl}^{(j)}$ as:

$$P_{kl}^{(j)} = \sum_{g \in G} d_{kl}^{(j)}(g^{-1}) T(g) \tag{39}$$

The algorithm to compute the desired coordinate system is now as follows:

1. Apply the operator $P_{11}^{(j)}$ to a set of basis vectors of the original vector space. The operator $P_{11}^{(j)}$ is a projection operator and the image of the basis vectors will contain c_j independent vectors.
2. Find these c_j independent vectors and apply the operators $P_{1k}^{(j)}, k = 2, ..., n_j$ to them. Each operator $P_{1k}^{(j)}$ will produce c_j new basis vectors from the ones constructed in the first step. This gives the $c_j * n_j$ basis vectors connected to the matrix τ_j.

B The learning rule

In our implementation we use a Newton based learning rule to find the maximum of the quality function. We recall that all the quality functions considered were of the type $Q(A) = \frac{Q_V(A)}{Q_C(A)}$ where $Q_V(A)$ was the determinant of the output covariance matrix and $Q_C(A)$ was a second function of the filter matrix A. The partial derivatives of Q are:

$$\frac{\partial}{\partial a_{lk}} Q(A) = \frac{\partial}{\partial a_{lk}} \left(\frac{Q_V(A)}{Q_C(A)} \right)$$

$$= \frac{Q_C(A)\frac{\partial}{\partial a_{lk}}Q_V(A) - Q_V(A)\frac{\partial}{\partial a_{lk}}Q_C(A)}{Q_C^2(A)}$$
$$= \frac{G(A)}{Q_C^2(A)}$$

Since we are interested in maximum values of the quality function we go in the direction of the gradient and we use the second derivative as the steplength parameter. This makes it necessary to compute the lk-th entries of the matrices G and $G'(A)$:

$$\begin{aligned} G(A(t)) &= Q_C(A(t))\frac{\partial}{\partial a_{lk}}Q_V(A(t)) - Q_V(A(t))\frac{\partial}{\partial a_{lk}}Q_C(A(t)) \\ G'(A(t)) &= \frac{\partial}{\partial a_{lk}}G(A(t)) \\ &= Q_C(A(t))\frac{\partial^2}{\partial {a_{lk}}^2}Q_V(A(t)) - Q_V(A(t))\frac{\partial^2}{\partial {a_{lk}}^2}Q_C(A(t)) \quad (40) \end{aligned}$$

The three different systems we considered used as Q_C the functions (see equations 16, 18, 24):

$$\begin{aligned} Q_C &= 1 \\ Q_C &= Q_2 = \sum_{i=1}^{N}\left[o_i^2\right](1 - \left[o_i^2\right]) \\ Q_C &= Q_4 = \sum_{k=1}^{K}\left[o_k^2 - o_k^4\right] \end{aligned}$$

In our update rules we need therefore the first and second derivatives of $Q_V(A)$, $Q_2(A)$ and $Q_4(A)$.

To compute the derivative $\frac{\partial}{\partial a_{lk}}Q_V(A)$ we note first that only the k-th column and the k-th line of the matrix $S = ([o_i o_j])$ depend on the weight a_{lk} :

$$\begin{aligned} Q_V(A) &= \left[o_k^2\right]\det S_{kk} + \sum_{i\neq k; j\neq k}[o_i o_k][o_j o_k](-1)^{i+j+\Delta(i,j,k)}\det\left(S_{kj}\right)_{ik} \\ &= \left[o_k^2\right]\det S_{kk} + \sum_{i\neq k; j\neq k}[o_i o_k][o_j o_k]\beta_{ij}^k \quad (41) \end{aligned}$$

where $\Delta(i,j,k)$ is defined as:

$$\Delta(i,j,k) = \begin{cases} 1 & \text{if } i > k \text{ and } j > k \\ 1 & \text{if } i < k \text{ and } j < k \\ 0 & \text{if } i > k \text{ and } j < k \\ 0 & \text{if } i < k \text{ and } j > k \end{cases} \quad (42)$$

S_{kk} denotes the $(N-1)\times(N-1)$ submatrix of $([o_i o_j])$ obtained by deleting the k-th row and the k-th column. $(S_{kj})_{ik}$ is the $(N-2)\times(N-2)$ submatrix of $([o_i o_j])$ obtained by deleting rows k and i and columns j and k. All these submatrices (and thus β_{ij}^k) are independent of a_{lk}. The first and second derivatives are computed as:

$$\frac{\partial}{\partial a_{lk}}Q_V(A) = 2[o_k p_l]\det S_{kk} + \sum_{i\neq k; j\neq k}\beta_{ij}^k\left([o_j o_k][o_i p_l] + [o_i o_k][o_j p_l]\right)$$

$$\frac{\partial^2}{\partial {a_{lk}}^2} Q_V(A) = 2\left[p_l^2\right] \det S_{kk} + 2 \sum_{i \neq k; j \neq k} \beta_{ij}^k \left[o_j p_l\right] \left[o_i p_l\right]$$

The first and second derivatives of $Q_2(A)$ are given by:

$$\begin{aligned} \frac{\partial}{\partial a_{lk}} Q_2(A) &= \frac{\partial}{\partial a_{lk}} \sum_{i=1}^{N} \left[o_i^2\right] \left(1 - \left[o_i^2\right]\right) \\ &= 2\left[o_k p_l\right] - 4\left[o_k^2\right] \left[o_k p_l\right] \\ \frac{\partial^2}{\partial {a_{lk}}^2} Q_2(A) &= 2\left[p_l^2\right] - 8\left(\left[o_k p_l\right]\right)^2 - 4\left[o_k^2\right] \left[p_l^2\right] \end{aligned}$$

and for the fourth order function we get:

$$\begin{aligned} \frac{\partial}{\partial a_{lk}} Q_4(A) &= \frac{\partial}{\partial a_{lk}} \sum_{i=1}^{N} \left[o_i^2 (1 - o_i^2)\right] \\ &= 2\left[o_k p_l\right] - 4\left[o_k^3 p_l\right] \\ \frac{\partial^2}{\partial {a_{lk}}^2} Q_4(A) &= 2\left[p_l^2\right] - 12\left[o_k^2 p_l^2\right] \end{aligned}$$