Technical Report

Pattern Recognition and Image Processing Group
Institute of Computer Aided Automation
Vienna University of Technology
Favoritenstr. 9/1832
A-1040 Vienna AUSTRIA
Phone: +43 (1) 58801-18351
Fax: +43 (1) 58801-18392
E-mail: langs@prip.tuwien.ac.at
URL: http://www.prip.tuwien.ac.at/

PRIP-TR-67 May 2, 2002

Irregular Eigenimage Pyramids and Robust Appearance-Based Object Recognition [1]

Georg Langs, Horst Bischof, Walter G. Kropatsch

Abstract

In this report we deal with the possibilities to improve the performance of robust PCA algorithms by replacing uniform subsampling of images by an irregular image pyramid. The image pyramid is built based on knowledge gained from the training set of images. It represents different regions of the image with different level of detail, depending on their importance for the reconstruction process. As this is a technical report, not only the final and optimal results will be presented, but also some of the previous steps will be explained, even if they were abandoned later. The main reason is to support future work on the topic.

[1]This research has been supported by the Austrian Science Fund (FWF) under grants P14445-MAT and P14662-INF.

1 Introduction

1.1 Motivation

Humans, when looking at each other's face, don't see every part at once, neither they try to cover the face with equaly distributed glimpses, but they concentrate on several special parts, like the eyes, the mouth or the ears. [14] and [15] propose two strategies to optain and apply information about the importance of different regions of an image when simulating the human visual system. Experiments during which an observer was asked to look at different modifications of the same image over different viewing sessions have shown that the scan paths of the eyes, called saccades, are not only dependent on a present picture, but also on previous knowledge.

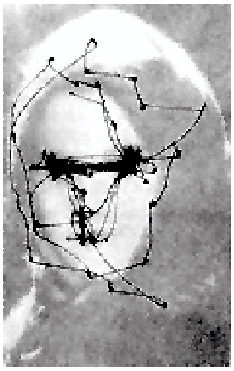


Figure 1: Tracks of the eye movements while looking at a face

In terms of *top-down* respectively *bottom-up* which describe abstract descriptions or internal cognitive-spatial models ([14]) respectively concrete (pixel values) information about an image, we can describe these strategies as follows:

1. Bottom-up methods retrieve their parameters only from the input image. Example: the input data is filtered by a gradient filter and only regions where the gradient value is above a certain threshold are used for recognition. Other features typically used are edges per unit area, entropy or local symmetry. ([16])

2. Top-down methods are driven by knowledge which is available before getting the input. In our case this will be the importance of pixels for recognition resp. reconstruction and the relations between pixels considering a training set of images. These parameters are used to design a process which will be applied onto an input image

This work deals with an application of the top-down strategy.

1.2 Robust Principal Components Analysis

Appearance-based methods consist of two stages: a training phase, during which a set of training images is obtained. An object usually is represented by images acquired under different viewing angles or illumination situations. These images are highly correlated. Thus a compression by principal component analysis (PCA) is highly efficient. [5, 1]
Each image is a point in a vector space. The coordinates are the pixel values. By changing the base of the vector space in order to adjust it to variance of the data, we can describe the data points sufficiently by a small part of the new base vectors which are the *eigenvectors* of the covariance matrix. The space spanned by the *eigenvectors* $e_1, ..., e_n$ is called the *eigenspace*.
In the eigenspace an object seen under different orientations forms a manifold (Figure 2 shows images of a rotating duck projected into the eigenspace, which is spanned by the first 3 eigenvectors. Because we can parameterize the rotation by one parameter (horizontal angle), the data points form a curve. If we would use two degrees of freedom (horizontal and vertical angle) the images would be projected on a surface in the eigenspace.) During

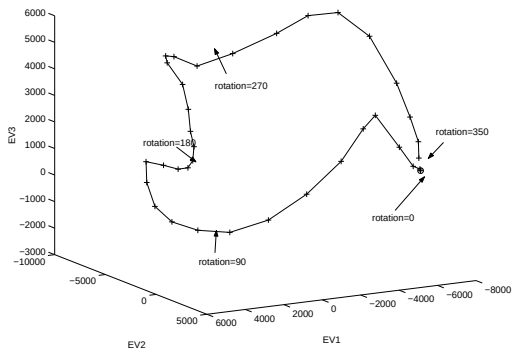


Figure 2: Projection into the eigenspace spanned by the first 3 eigenvectors.

the recognition phase the input image is projected into the eigenspace of the training set and the closest point on the manifold is considered a valid match if the distance does not exceed a certain threshold.
Instead of projecting the whole input image into the eigenspace, robust PCA coefficients of the projection are determined by a robust hypothesize-and-test paradigm. We approximate an image only by a linear combination of a subset of eigenimages i.e. we use a subspace of the eigenspace, which is spanned by a certain number of eigenimages $e_1, ..., e_p$ with $p < n$. Based on randomly chosen subsets of pixels in the input image, a set of hypotheses is built. The subsets have size $k > p$. Each hypothesis (i.e. set of coefficients) is the solution of an overconstraint system of equations in the least squares sense based on k datapoints. To cope with outliers, the solutions are calculated in a robust way: again subsets of the k data points are selected and iteratively reduced according to the error distribution.
Competing hypotheses i.e. sets of eigenvector coefficients, are selected according to the

Minimum Description Length principle.[1, 5]
Outliers, noise or occlusions are likely to be excluded from coefficient reconstruction, because they are either eliminated during robust solution of the equation system or excluded with bad hypotheses.

1.3 Irregular Pyramids

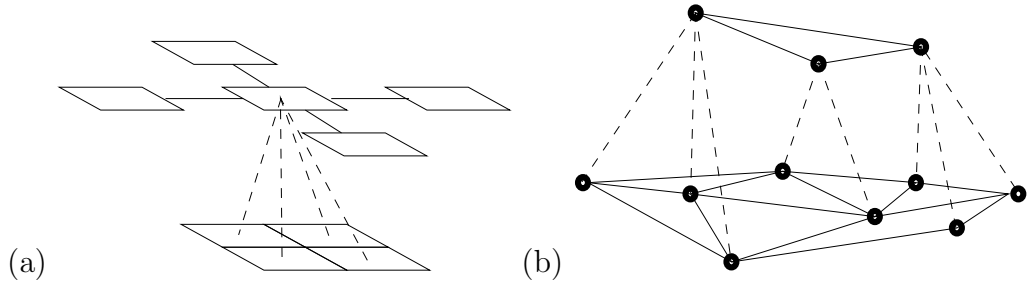


Figure 3: Father-son relations in regular (a) and irregular (b) pyramids.

While regular image pyramids are usually a stack of versions of one image that differ by their gradually reduced spatial resolution, irregular pyramids operate on a general graph structure and are therefore capable of much more complex tasks. For detailed introduction to the subject see [6, 7, 8], for further information about some algorithmic and theoretical concepts see [9, 12, 17, 13].
The structure of a pyramid is determined by 1. neighbour relations within one level and 2. father-son relations between adjacent levels (For a detailed description see [8]). Except in the base level, every node (father) is assigned a set of nodes (sons) in the level directly below that provide its input. Figure 3 depicts such father-son relations in the case of regular (a) and irregular pyramids (b).
Each node is assigned a set of attributes. In case of multiresolution images they are grey values, whereas other pyramids may use multiple and more abstract attributes, for example the shape of the receptive field etc.
For the construction of irregular eigenimage pyramids the attributes of the nodes were the mean *weight* of the receptive field and the *value profile* as described below.

1.4 Our approach

The approach presented in this report aims to support robust recognition based on eigenimages ([1]). We deal with an input, which can be a picture or any other signal, that is represented as a vector of pixel values. Instead of performing saccades we change this representation in order to stay abreast of the regions of interest and build image pyramids.

The algorithm will proceed with the following major steps:

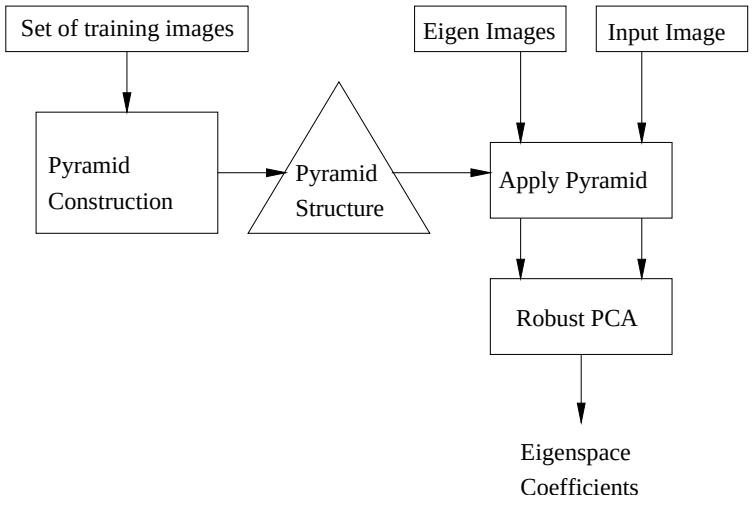


Figure 4: The basic concept of our algorithm. It is divided into a training- (left) and a recognition phase (right).

- teach the computer the importance of certain parts of images as well as the dependencies of different parts, given a set of objects to recognize or reconstruct resp.
- enable the computer to apply this knowledge during the recognition and segmentation process, by building image pyramids and applying them on input images.
- put as much computational expensive steps as possible in the offline phase.

Applying the pyramid structure on the images supports the robust appearance based recognition in the following way:
Regions with small importance for the reconstruction or recognition are usually similar on different images of the training set, therefore if used for robust recognition they support almost all hypotheses. Robust recognition seeks for pixels, that are consistent with a given hypothesis, a large set of irrelevant pixels that are consistent with almost all hypotheses strongly interferes with the method in three ways:

1. It causes huge equation systems in the fitting step,
2. it wastes time because useless hypotheses are built.
3. difference between good and bad hypotheses is likely to become smaller.

Our approach is divided into 2 main phases: (Figure 4)

1. training phase: building of the pyramid structure depending on knowledge based on the *training set* of pictures or vectors, one has to recognize.

2. recognition phase: different levels of the pyramid structure are applied on the input picture as well as on the eigenimages of the database. It starts with higher levels, that contain less information, and continues to lower levels only if results are not 'secure' enough.

1.5 Overview of this report

The technical report is organized as follows: section 2 describes how the pyramid structure is built. Section 2.2 gives a definition of the pyramid structure, in section 2.3 two methods of contraction are introduced and analysed. The application of the pyramid structure during the reconstruction phase is explained in section 3. In section 4 experiments on the construction of the pyramid (4.1) and reconstruction (4.2) are presented and discussed. In the conclusion (5) we give a a short summary of our results and present an outlook and suggestions for future work on the topic.

2 Training Phase

2.1 Using available knowledge

We assume that we are given a set of images that represent one or more objects respectively their orientations. Furthermore we assume that we are going to use only one eigenspace to recognize this set of images later on. All images are of the same size. This gives us the possibility to retrieve the following data:

1. the *variance matrix*, which represents the variance in each pixel over the training set of images

2. eigenimages of the training set

3. dependencies between pixels i.e. correlation between the values of two neighbouring pixels over the training set: For a given pixel i the pixel values in each image of the training set form a vector of values $(v_{i,1}, ..., v_{i,n})$. We call this vector the *value profile* of a pixel position. Two such vectors can be compared by calculating their correlation $corr(v_i, v_j)$. In levels where we deal with graphs instead of images the *value profiles* are defined on the nodes of the graph. The value of a node is calculated by to a reduction function wich still has to be defined. During this work we calculated the values of a nodes by building the mean value of their receptive fields in the base level of the pyramid. Figure 5 visualizes *value profiles* for levels with regular (a) (i.e. the image) and irregular (b) graph structures.
   By contracting two nodes with highly correlated value profiles the loss of information is expected to be smaller than the loss caused by contracting two pixels with more independent behavior.

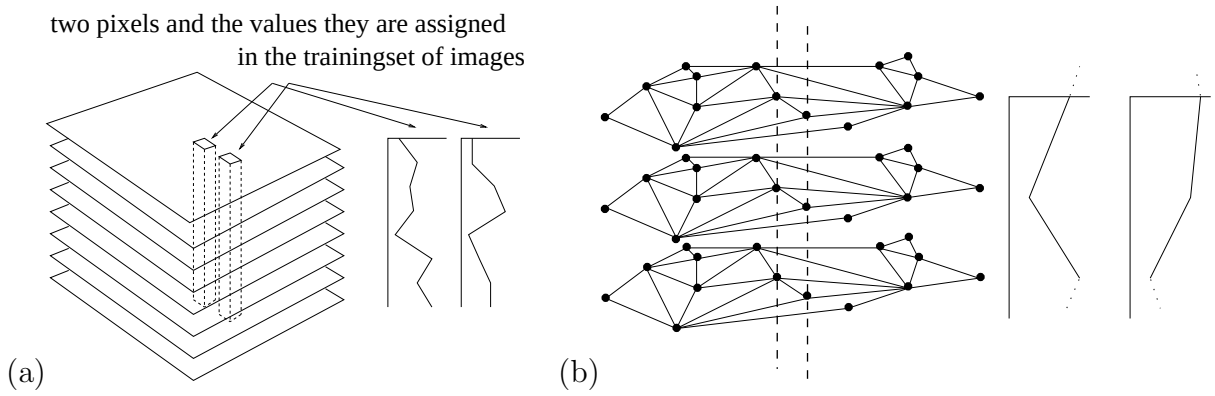


Figure 5: Value profiles of neighbouring nodes in the training set

Based on the *eigenimages* and the *variance matrix* we retrieve a *weight map* $:= (\omega_i)_{i=1,...,m}$ where $0 \leq \omega_i \leq 1$. Figure 6 shows a weight map based only on the normalized values of the variance matrix. It is based on 36 images of a duck that was rotated around its vertical axis. The x- and y-values represent the coordinates of the pixels in the image, the z-value represents the variance. Note the high variance values in the region where the head is located at the area with y-value $0 \leq y \leq 55$. To make orientation easier, one image of the data set is depicted below the graph of the weight map.

The construction of the *weight map* has strong influence on the resulting pyramid and the performance during recognition. It will be discussed in detail later. Based on the *value profiles* of the training set a *distance matrix* d_{ij} is calculated. It represents a distance function between two nodes or pixels. The closer two nodes are, the less information is lost if their receptive fields are merged i.e. if they are contracted. In section 4 pyramids resulting from different functions $d_{i,j}$ are shown. We will use the correlation coefficient of the value profiles ($v_{i,k}$ and $v_{j,k}$ (where k is the index of the image in the training set $(x)_k$ with $k = 1, ..., n$ and i, j are the indices of two pixels) of two nodes, and let $d(i, j) = 2 - corr(v_{i,k}, v_{j,k})$. Because only distances between neighbouring nodes are used for the contraction, it is computed during construction of the pyramid. The complexity of the step is reduced from $O(n^2)$ to $O(n)$.

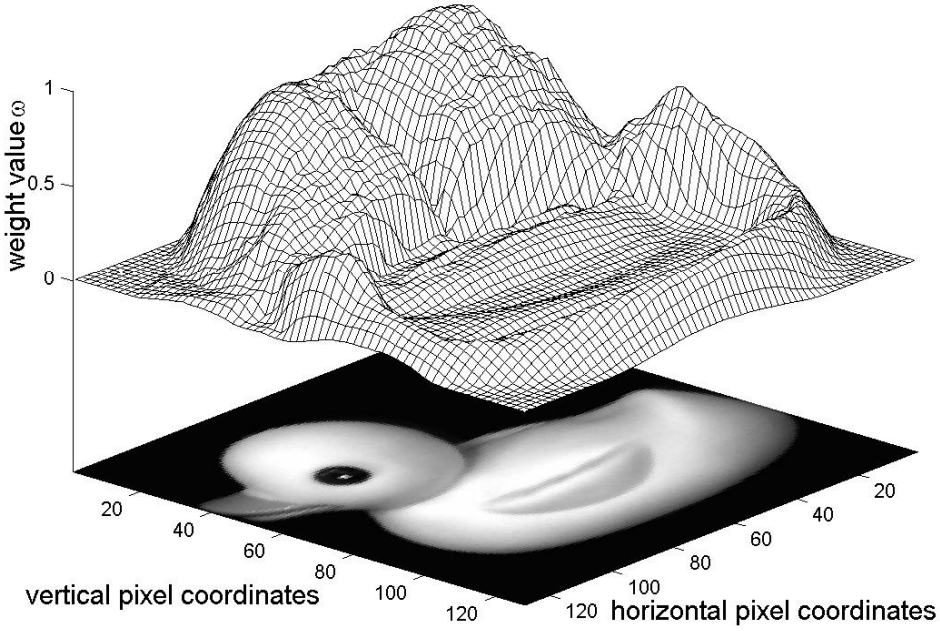


Figure 6: weight map based on variance for a training set, consisting of 36 images of a rotating duck.

2.2 Pyramid structure

The result of our algorithm is a *pyramid structure* that can be applied to any input image of a given size. In 2.2.1 we describe the structure and give its exact definition, in 2.2.2 we explain an example.

2.2.1 Definition of the pyramid structure and its application

The pixels of the input image can be indexed by $i = 1, ..., N_1$ where N_1 is the number of pixels in the image.
To convert a rectangular grid (the image) to a vector we used the following transformation where i_{vector} indicates the index in the vector and (i_{array}, j_{array}) the vertical and horizontal coordinates of a pixel in an $m \times n$ image.

$$i_{vector} = (i_{array} - 1)n + j_{array} \tag{1}$$

Each pyramid level P_k consists of a vector of nodes, each of them representing a set of pixels, its *receptive field*.

$$P_k = \langle n_{1,k}, ..., n_{N_k,k} \rangle \tag{2}$$
$$\forall i, k : n_{i,k} \in \mathfrak{P}(\{1, ..., N_1\}) \tag{3}$$

where

$$n_{i,k} \cap n_{j,k} = \emptyset, i \neq j \qquad \text{and} \qquad \bigcup_{i=1,...,N_k} n_{i,k} \cup r = \{1, ..., N_1\} \tag{4}$$

$$r = \{i : \omega_i = 0\} \tag{5}$$

i.e. the receptive fields are not overlapping each other and the union of the receptive fields together with r covers the whole image. r is the set of pixels, that have $weightmapvalue = 0$. They are irrelevant or interfering with recognition and therefore are ignored.
In the first pyramid level each node represents one pixel:

$$n_{i,1} = i \tag{6}$$

During contraction a node in the level $k+1$ takes over the indices from its sons in level k:

$$n_{i,k+1} = \bigcup_{j:n_{j,k} \text{son of } n_{i,k+1}} n_{j,k} \tag{7}$$

The final pyramid structure with L levels consists of L vectors of nodes $P_k, k = 1, ..., L$, each representing a receptive field in the base level.

Now we will define the procedure how to apply an empty pyramid structure P on input-data: To ease reading, from now on we assume the data to be an image, still the concept can be applied in a straight forward way to any input representable in vector form. The structure can be applied to an input image independently for each level, i.e. one can construct a certain level of the pyramid directly from the input image without constructing the levels in between.

Definition 2.1 *Let B_1 be an image with pixel values $\langle b_{i,1}\rangle_{i=1,...,N_1}$ (in our experiments we used an image size of $N_1 = 128 \times 128 = 16384$ pixels). To* ***calculate the k-th level B_k of the pyramid****, for each node the mean value of the pixel values in the receptive field is built:*

$$B_k = \langle b_{1,k}, ..., b_{N_k,k}\rangle \tag{8}$$

$$\forall i : b_{i,k} = \frac{\sum_{j \in n_{i,k}} b_{j,1}}{|n_{i,k}|} \tag{9}$$

Note that P is an 'empty' structure, the nodes don't have attributes assigned to them. Only when calculating B_k of an input-image, attributes like gray values are assigned to the nodes of B_k according to P_k.

2.2.2 Example of a pyramid structure

To illustrate the pyramid structure we give an example of a level P_k and the calculation of the according level B_k: Figure 7 shows two examples of levels of the pyramid structure introduced above. (a): A base-level is depicted. The size of the images in the training set is

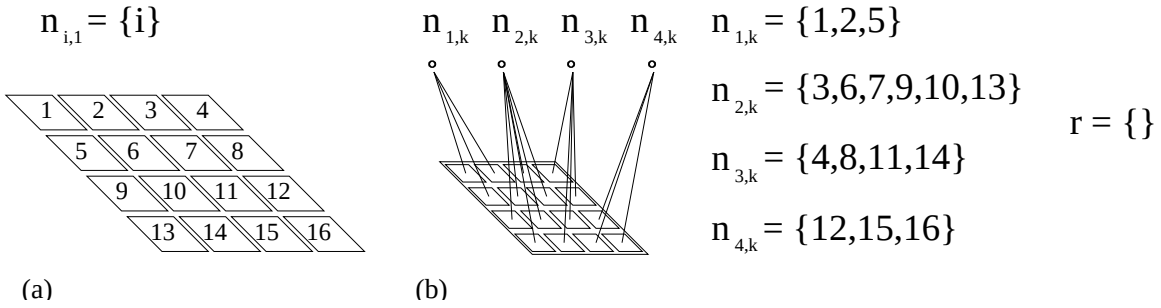


Figure 7: An example of (a) the base-level and (b) a pyramid level P_k with 4 nodes.

$4 \times 4 = 16$ and for all nodes in the base-level $n_{i,1} = i$ (calculated according to equation 1) holds. (b) shows a level with 4 nodes, each of them representing a set of pixels in the base-level. The set r is empty in this example.

Figure 8 shows an example of an 4×4 input-image. Given level k from figure 7 we calculate B_k according to definition 2.1:

$$B_1 = \langle 1, 4, 2, 3, 3, 8, 3, 6, 8, 9, 9, 4, 7, 5, 3, 5\rangle \tag{10}$$

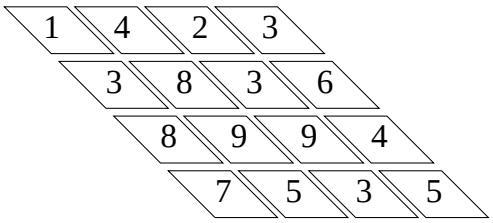


Figure 8: An example of an input-image

Every node $n_{i,k}$ defines a value $b_{i,k}$:

$$b_{1,k} = \frac{1+4+3}{3} = 2.6667 \tag{11}$$

$$b_{2,k} = \frac{2+8+3+8+9+7}{6} = 6.1667 \tag{12}$$

$$b_{3,k} = \frac{3+6+9+5}{4} = 5.75 \tag{13}$$

$$b_{4,k} = \frac{4+3+5}{3} = 4 \tag{14}$$

and the image is represented by the vector

$$B_k = \langle 2.6667, 6.1667, 5.75, 4 \rangle \tag{15}$$

2.3 Contraction

During the contraction process we represent a given image with decreasing precision. Basically this is achieved by decreasing the resolution of the image. The traditional subsampling process works with a gaussian pyramid [6]. All regions of an image are treated the same way.
Our pyramid is generated in a similar way, but based on previous knowledge we vary the loss of information resp. the resolution in parts of the image in contrary to the traditional algorithm. This strategy leads to an irregular pyramid, where we no longer deal with regular grids, but with connected regions, that can have arbitrary shape. The subsampling process is realized by merging the regions.
To minimize the loss of information, we enforce the merging of regions with high correlation, and try to avoid merging of regions with directly opposed value profiles throughout the training set.

Our algorithm deals with the following questions:

1. Divide the nodes in a given level into *survivors* and *non survivors* i.e. decide whether a certain receptive field is merged with another receptive field when constructing the successor of the level

2. if a node does not survive, decide which neighbouring *survivor* shall become its father. The receptive fields are merged.

3. If the resulting receptive field does not meet certain requirements, change the status of the *non survivor* to *survivor*.

The answer to point 1 is based on the stochastic decimation algorithm by Peter Meer. [17] A threshold on an attribute (weight) of a node could decide whether a node survives or not, too. This criterium would have the following drawbacks:

1. It does not build receptive fields

2. Relations between neighbouring pixel positions don't influence the contraction.

The reason why we use stochastic decimation is the following: for a proper pyramid structure (see definition in [8]) we need a set of receptive fields, that cover the whole image. Furthermore every pixel has to have a father, i.e. a node, representing this pixel in higher levels although the pixel itself is a nonsurvivor. Stochastic decimation solves this task by building a maximum indepedent set of surviving nodes or pixels:

1. Each non-survivor has at least a surviving neighbour.

2. No survivor has a surviving neighbour.

Our modification ensures condition 1, while condition 2 is not required in order to adjust the pyramid to the given *weight map*. (in [10] an opposite approach is described. To increase the speed of convergence the 1^{st} condition is omitted. Time is not crucial for our application because the construction of the pyramid takes place in the offline phase, furthermore we make use of a large number of levels which yields a higher variety of different resolutions).
In the first step this algorithm assigns every pixel i.e. node a random value. Local maxima become survivors, their neighbours loose the possibility to survive. Iteratively survivors and candidates, which are nodes which have not been chosen to be survivors nor non-survivors yet are assigned a new random value. Again local maxima are chosen to survive while their neighbours 'die'. This procedure is repeated until a maximum independent set of surviving nodes is reached, that is a set of nodes, where no survivor has a surviving neighbour, but every non-survivor has at least one surviving neighbour. We modify this algorithm in a way, that gives nodes which are stochastic non-survivors the chance to survive depending on a *weight* $0 \leq \omega_i \leq 1$ that is assigned to pixel i by the *weightmap*. We can do that in different ways, that we will present and compare in the following:

Dependent distribution contracion:
After performing stochastic decimation every non - survivor survives if a random number $0 \leq r \leq 1$ is less than $f(\omega_i)$. We define f by a parameter *survexp*: $f(\omega_i) = \omega_i^{survexp}$. The higher *survexp* is, the higher are the chances for a pixel

to survive despite stochastic decimation. The set of surviving nodes is no longer independent, but is distributed dependent on the *weightmap*. Merging of the receptive fields i.e. the search for a father is controlled by a function $d((x_i)_k, (x_j)_k, \omega_i, \omega_j)$ (in the following referred to as $d(i,j)$) depending on the differences between the weights of neighbouring pixels and their correlation: a non-survivor v_i chooses the surviving neighbour x_j to be its father, if $d(i,j)$ is maximal for
$\{j : v_j$ neighbour of $v_i\}$, $x_j$is survivor.
We are able to control 2 parameters: *survexp*, a function $d(v_{i,k}, v_{j,k}, \omega_i, \omega_j)$ (in the following referred to as $d(i,j)$) depending on the differences between the weights of neighbouring pixels and their correlation and the *weightmap*. *survexp* defines the dependence of the subsampling process on the *weightmap*. d provides a rule for merging receptive fields.

Weight target contraction:
After performing stochastic decimation, all non-survivors with weight above a certain limit become survivors. The limit is $1 - \tau$. The parameter τ denotes the *target tolerance*.
In levels $2, 3, ...$ we deal no longer with pixels, but with nodes that represent a set of pixels in the original image or weight map. A node $n_{i,l}$ in level l is assigned the weight

$$\omega_{n_{i,l}} = z \cdot \sum_{x_i \in \text{ receptive field of } n_{i,l}} f(\omega_i) \tag{16}$$

z is a factor initialized with 1 and decreased during contraction in order to ensure convergence to one single node. f is the *weight contraction function*. It will be discussed in detail later. There is either no decrease or the rate of decrease is 0.5 according to step 5 of the algorithm. After we have chosen a set of surviving nodes, again we have to assign every non-survivor a *father*. All *sons* of one *father* build its receptive field. P_1 is the base level of the pyramid and its neighbourhood relation $\hat{N}_1 \subseteq P_1 \times P_1$ defined according to the input data. If level P_i with the neighbourhood relation $\hat{N}_i \subseteq P_i \times P_i$ has been built then we build level P_{i+1} according to the following rules:

1. perform stochastic decimation on P_i thus every node is assigned a status *survivor* or *non-survivor*.
2. a *non-survivor* n_{i,j_0} chooses a father from all neighbouring survivors $\{n_{i,j_1}, ..., n_{i,j_k}\}$. $n_{i,j_{father}}$ becomes father if its *value profile* is most correlated with the *value profile* of the *non-survivor* $n_{i,j}$ i.e. if the distance $(d_i)_{j_0,j_{father}}$ is minimal. d_i is the *distance map* of level i
3. if the *weight* of the *non-survivor* $\omega_{n_{i,j_0}}$ and the *weight* of the chosen *father* $\omega_{n_{i,j_{father}}}$ sum up to a value

$$\omega_{n_{j,j_0+1}} = \omega_{n_{i,j_0}} + \omega_{n_{i,j_{father}}} > (1 + \tau) \tag{17}$$

then do not merge and change status n_{i,j_0} to *survivor*

4. define the neighbourhood relation of the new level according to stochastic decimation [17].
5. If contraction terminates set $z_{new} = z/2$.

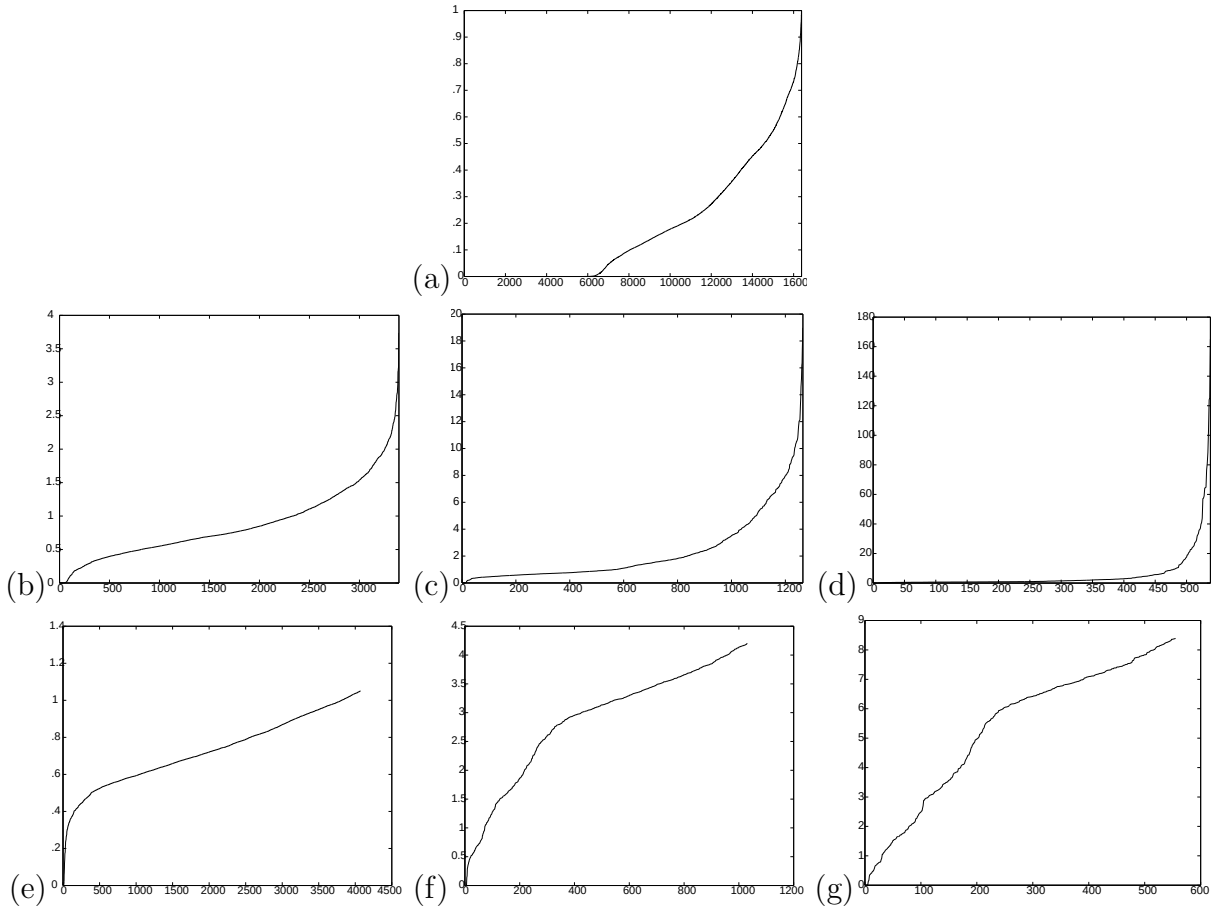


Figure 9: Distribution of weightvalues on the nodes of different levels

Weight target contraction deals with the distribution of weights on the set of nodes on one level (figure 9). While its first priority is still to merge receptive fields with high correlation it merges them only until they reach a certain weight according to (17). This weight is initialized with the maximum of the *weightmap*. The contraction process proceeds until no further merging of two receptive fields is possible without generating a receptive field with a weight above a tolerance limit which is for example 10% above the *weighttarget*. The contraction converges and finally stops completely. At this point z is decreased and again contraction is performed until it converges. This strategy leads to a better balanced distribution of weights compared to DD-contraction. Figure 9 shows the distribution of weights of receptive fields achieved with the two methods. In Fig.9 (a) the distribution in the original image is depicted.

Fig.9 (b,c,d) and Fig.9 (e,f,g) show distributions after contraction with DD- and WT-contraction respectively. For DD-contraction the levels (b) 3, (c) 5 and (d) 8 with a numbers of nodes 3390, 1265 and 538 were analysed. For WT-contraction we analysed levels (e) 8, (f) 14 and (g) 17 with 4073, 1031 and 666 nodes.

In contrary to contraction with Hopfield networks [2, 3, 11], in our algorithm selection is influenced first by the *distance function* (search for father) and then by the resulting *weight* (whether to survive or not). This corresponds to a priority for small distances between contracted nodes. Hopfield networks perform selection according to an energy function which has to be minimized. They operate on a neighbourhood graph where weights ω_{pq} are assigned to edges between neighbouring nodes p and q. The nodes have states $s(p) \in \{0, 1\}$. A valid decimation is computed with an update procedure for node states:

$$s(p) = \begin{cases} 1 & \text{if } I_p + \sum_{(p,q)\in N} \omega_{pq} s(q) > 0 \\ 0 & \text{otherwise} \end{cases}$$

Here distance (ω_{pq}) and value of a node enter the algorithm in the same step.

Experiments (2) show that with $f(\omega_i) = \omega_i^s$ in (16) there is no exponent s that performs best on all resolutions resp. levels. Therefore we replace the term ω_i^s by a more general term to gain better control of the development of receptive fields trough out the pyramid.

Theorem 2.1 *Let* $(\omega_i)_{i=1,...,N}$ *be a* weight map *with* $0 \leq \omega_i \leq 1$ *for all* $i = 1, ..., N$. *Let* P_i *denote the set of nodes and* $\hat{N}_i \subseteq P_i \times P_i$ *be the neighbourhood relation in level* i. *Let* d_i *be* distance maps *i.e. functions from* N_i *in* $[-1, 1]$.
Then WT-contraction converges to a single node.

Proof 2.1 *If there is more than one node after finishing the iteration i.e. if* $|P_i| > 1$, *there are two cases:*

1. *No further contraction took place in the last step (* $|P_{i-1}| = |P_i|$ *), hence the factor* z *was decreased to* $z = z_{old}/2$, *for all receptive fields* $n_{i,l}$ *the following equation holds:*

$$\omega_{n_{i,l}} = \left(\sum_{x_i \in \text{ receptive field of } n_{i,l}} z \cdot f(\omega_i) \right) \leq \frac{1 + targettolerance}{2} \tag{18}$$

hence after performing stochastic decimation, during step 3 of our algorithm no status is changed from non-survivor to survivor, because all surviving neighbours of an arbitrary non-survivor have a value $\omega_{n_{i,l}} \leq (1 - targettolerance)$, *hence the algorithm contracts like stochastic decimation in this step. Convergence of stochastic decimation is proven in [17].*

2. *Contraction took place in the last step, therefore either contraction is possible or if* $z \neq z_{old}/2$ *and case 1 becomes valid in the next step.*

□

In addition it is possible to estimate the size of a receptive field, if we are given an interval, the weights of the pixels lying in the receptive field. This is helpful when one searches a function $f(\omega_i)$:

$$f(\omega_i) : [0,1] \rightarrow [0,1]\,, \text{monotonously ascending} \tag{19}$$

Let $n_{l,j}$ be a node with a receptive field $\{p_i, i = 1, ..., N\}$ and let ε denote the targettolerance, then

$$\omega_{n_{l,j}} = z \sum_{i=1}^{N} f(\omega_i) \leq (1+\varepsilon) \tag{20}$$

holds. If we assume that

$$\forall i = 1, ..., N : \omega_i \in [\bar{\omega} - \delta, \bar{\omega} + \delta] \tag{21}$$

and reformulate equation (20), we get the estimations

$$N \cdot f(\bar{\omega} - \delta) \leq \textstyle\sum_{i=1}^{N} f(\omega_i) \leq N \cdot f(\bar{\omega} + \delta) \tag{22}$$

$$N \leq \tfrac{\frac{1}{z}(1+\varepsilon)}{f(\bar{\omega}-\delta)} \tag{23}$$

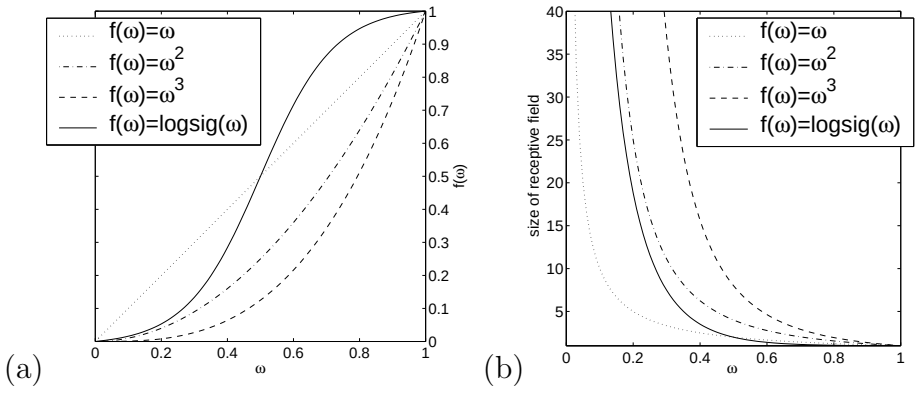


Figure 10: (a): Some functions $f(\omega_i)$ (ω_i^s, $logsig(\omega_i)$), (b): a comparison of the corresponding sizes of the receptive fields.

Figure 10 gives an impression of the influence of $f(\omega_i)$ on the size of the receptive fields. The function *logsig* is a modified log-sigmoid transfer function. We first define

$$ls(\omega) = \frac{1}{(1 + e^{-\omega})} \tag{24}$$

Now we modify ls in order to define the function $logsig : [0, 1] \rightarrow [0, 1]$

$$logsig(\omega) = \frac{ls(l \cdot (\omega - t)) - ls(-l \cdot t)}{ls(l \cdot (1 - t)) - ls(-l \cdot t)} \tag{25}$$

l and t are parameters. $l \in (0, \infty)$ controls the steepness of the curve, while $t \in [0, 1]$ shifts the steepest part of the curve along the x-direction.
If the weights are an equally distributed set, we can give an estimation of the influence of receptive fields on the final representation. We define this influence by the relative size of the set of nodes, that represent a region of a certain weight interval in the vector generated by the pyramid. For an experimental comparison of the methods see section 4.2, item 2.

3 Reconstruction

The following steps reconstruct or recognize a given image using a given eigenspace with a base consisting of N eigenimages, a pyramid P and pyramid level i. P is an empty structure as it is described in section 2.2. In particular we calculate B_i according to definition 2.1:

1. for all eigenimages of the eigenspace: calculate level B_i according to the pyramid P. This results in N vectors $\{e_1, ..., e_N\}$

2. calculate pyramid level $B_i^{inputimage}$ of the input image

The coefficients of the training images and the input image do not change [5]. The resulting vectors $\{e_1, ..., e_N, B_i^{inputimage}\}$ are input to robust PCA. Note that the 1^{st} point is performed during the training phase. During reconstruction only one level based on the input image has to be calculated. Computational expensive steps i.e. the contraction of an image template to a pyramid structure takes place entirely during the training phase.

4 Experiments

Experiments were performed on a dataset of graylevel images of different objects. The database (Columbia Object Image Library COIL-20 [18]) contains images of 20 objects, each object rotated around its vertical axis with images taken in 5^o steps. Our training set consisted of 36 images (i.e. 10^o degree steps) of one object taken from the database. The eigenspace used for reconstruction was spanned by 36 eigenvectors.

4.1 Building of the pyramid

In this section we present some results of our pyramid building algorithm, based on a training set consisting of 36 images of a duck[18]. Figures 11, 12, 13 and 14 compare different possibilities for $d(i,j)$ and *survexp*. In each figure the images (a) and (c) show randomly colored receptive fields of a low (a) and a high (c) level. Images (b) and (d) show the result of calculating B_i of an input-image and backprojecting the attributes of the nodes onto their receptive fields. (b) and (d) correspond to (a) and (c) respectively:
In figure 11 the distance function, used by DD-contraction, was defined by the difference

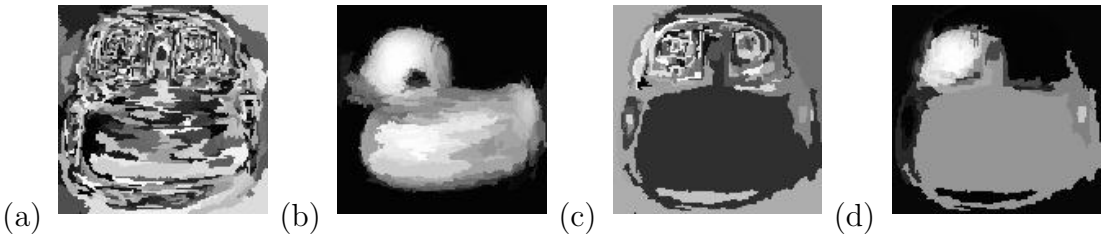


Figure 11: DD-contraction, (a,b) level 6 and (c,d) level 15, $survexp = .5$, $d(i,j) = -abs(\omega_i - \omega_j)$

of the weights of two neighbouring nodes. This strategy was motivated by the observation, that large receptive fields in unimportant regions 'swallow' smaller receptive fields in more important neighbouring regions. Merging two receptive fields with significantly different sizes almost completely overwrites the information in the smaller field by information in the larger field when the pyramid is applied to an image (See definition 2.1). Of course this interferes with our goal to represent regions according to their importance. Choosing $d(i,j) = -abs(\omega_i - \omega_j)$ avoids this development, but results in receptive fields, that spread over regions with independent pixel values i.e. poorly correlated *value profiles*, too. This causes a loss of information.
We combined correlation and difference of weights in the *distance function* to avoid this loss. The results are depicted in figure 12 with $d(i,j) = corr(i,j)^2 - (\omega_i - \omega_j)^2$ and figure 13 with $d(i,j) = corr(i,j)/abs(\omega_i - \omega_j)$. Note that receptive fields in are as 'compact' as in figure 11 but are divided in smaller subfields. Particularly large horizontal fields are avoided. The duck in our training set was rotated around a vertical axis, therefore the correlation between horizontal neighbours is usually less than with vertical neighbours.

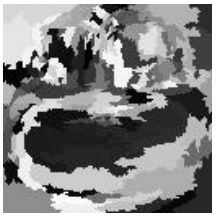
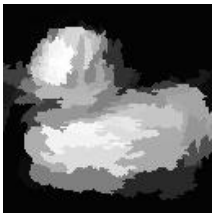
(a)  (b)  (c)  (d) 

Figure 12: (a,b) level 4 (c,d) level 6, $survexp = 2.5$; $d(i,j) = corr(i,j)^2 - (\omega_i - \omega_j)^2$

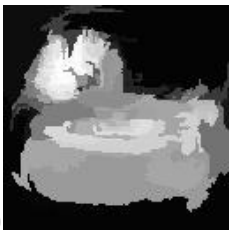
(a)  (b)  (c)  (d) 

Figure 13: (a,b) $survexp = 3$ and (c,d) $survexp = 1$; $d(i,j) = corr(i,j)/abs(\omega_i - \omega_j)$

Figure 14 shows results of a distance function defined entirely by correlation of *value profiles*. This definition seemed to be the most natural choice, and resulted in the best recognition rate. Figure 14 (a) and (b) show receptive fields with $survexp = 3$, (c)and (d) with $survexp = 1$. Generally smaller exponents $survexp$ result in more significant differences in the size of receptive fields, due to the algorithm described in section 2.3, Dependent distribution contraction.

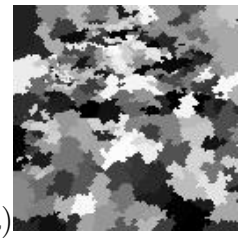
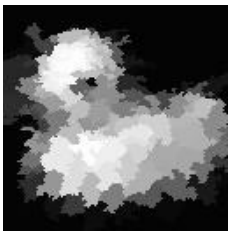
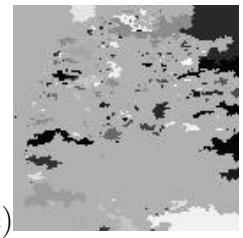
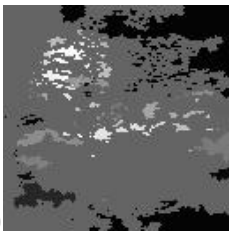
(a)  (b)  (c)  (d) 

Figure 14: (a,b) $survexp = 3$ and (c,d) $survexp = 1$; $d(i,j) = corr(i,j)$

Figure 15 shows receptive fields of a pyramid constructed with WT-contraction. Levels with (a) 117, (b) 1031 and (c) 2015 are depicted, For our experiments we used $targettolerance = 0.1$, i.e. all pixels resp. nodes whose weight is above 0.9 survive.

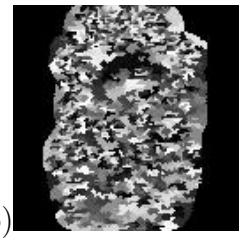
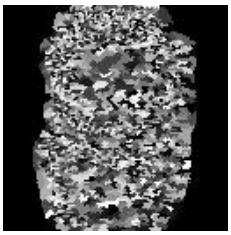
(a)  (b)  (c) 

Figure 15: Weight target contraction, $f(\omega_i) = \omega_i$, $d(i,j) = corr(i,j)$

4.2 Reconstruction

In the experiments the algorithm had to deal with images (which were taken from Columbia Object Image Library COIL-20 [18]) each of which was 50 percent occluded. The images were centered i.e. the mean image of the training set was subtracted. Therefore the image values are between -255 and 255 (The interval results from a dataset of 8 bit input images, that was centered around the origin of the image space by subtracting the mean image, further calculations were carried out with *double precision*). The images of one object differ only by the vertical rotation angle, therefore a weight map based on the variance of the pixel values is symmetric. This property ensures that by setting the right half of each image to 0, approximately 50% of the nodes are occluded for all levels . Regions with different weights are occluded to the same extend (with slight variations caused by the pyramid, that is not exactly symmetric) (Figure 16). For all levels of the pyramid each input image was reconstructed after preprocessing by one of the three methods:

- Downsampling using a Gaussian pyramid
- DD-contraction, with different values for parameter *survexp*
- WT-contraction, with different weight contraction functions $f(\omega_i)$

To compare the results we calculated the mean squared reconstruction error of the resulting 36 images for each level. To understand the results better, we studied the connection between pixel *weights* and the *size* of receptive fields on different levels. When studying levels of different pyramids, we compared levels with most similar number of nodes.

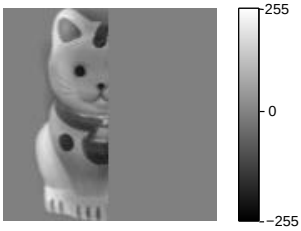


Figure 16: The test set consists of centered images that were 50 percent occluded

1. **Reconstruction with lower pyramid levels**
   With lower pyramid levels *WT-contraction* generally outperforms *DD-contraction*. Figure 17 shows a comparison of the reconstruction results for regular and irregular downsampling pyramids constructed with *WT-contraction*. For reference (a) shows an image reconstructed with full resolution ($128 \times 128 = 16384$ pixels) and no occlusion, (b) shows the reconstruction result based on the 6^{th} level of the irregular pyramid. The level had 543 nodes $\sim 3.3\%$ of the original size of the image, (c) shows the reconstruction result from a regular pyramid with $23 \times 23 = 529$ pixels.

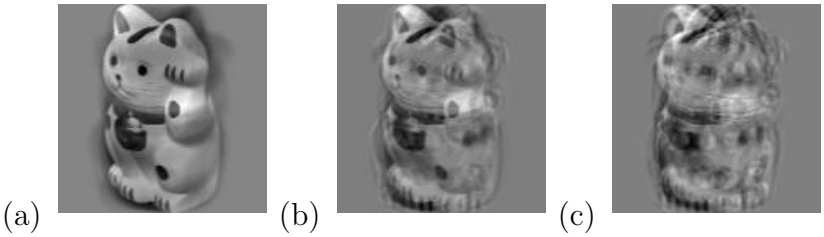


Figure 17: Reconstruction results with (a) full resolution, (b) irregular (543 nodes) and (c) regular pyramid (529 nodes)

For qualitative evaluation figure 18 shows the absolute differences between the reconstruction results and the ideal reconstruction with full resolution and no occlusion: figure 18(a) for an irregular pyramid, (b) for a regular pyramid. Figure 18 (c) and (d) show the corresponding pixel error histograms. The data used was the same as for figure 17.

For WT-contraction experiments with different exponent s in $f(\omega_i) = \omega_i^s$ (reconstruction errors are depicted in figure 19) indicate the following behaviour: For $f(\omega_i) = \omega_i^s$ at lower levels (i.e. smaller receptive fields) lower values for s outperform higher values. This corresponds to smaller receptive field sizes at higher weights as predicted in figure 10(b). Figures 21(c) and (d) show the relation between the *weight* ($\omega \geq 0.7$) of a pixel and the size of the receptive field within it lies. The respective data was derived pyramids constructed during our experiments: (c) from the 4^{th} level of a

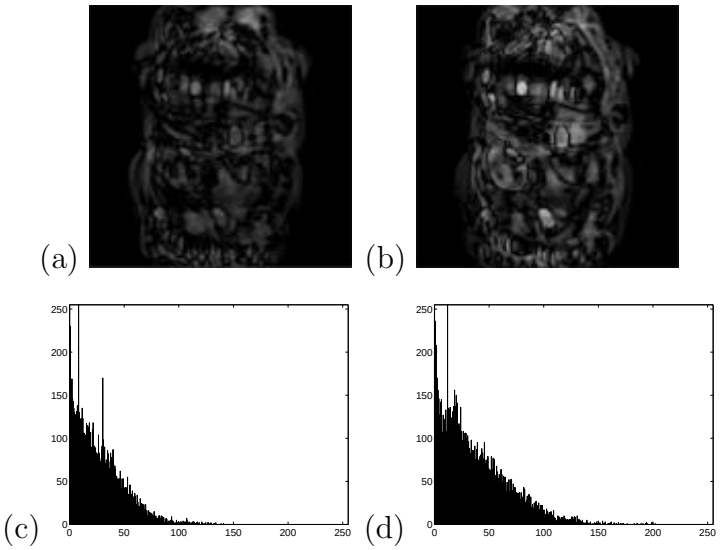


Figure 18: Reconstruction error for (a) irregular and (b) regular pyramid. Histograms of the error images (c) irregular and (d) regular pyramid

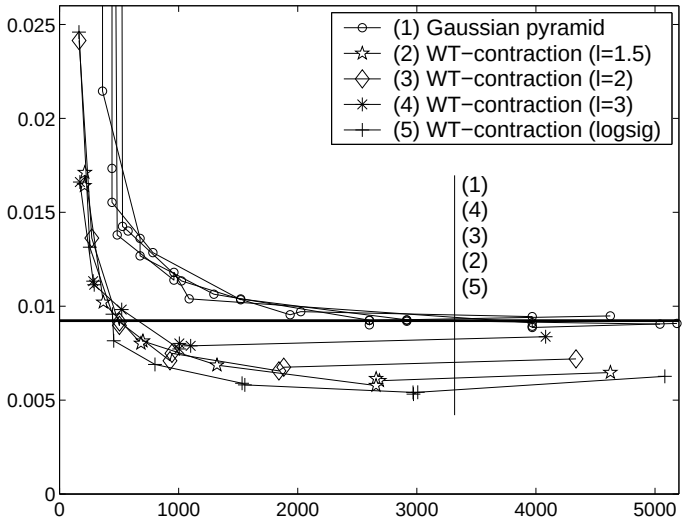


Figure 19: With a number of nodes above ≈ 400 a modified logsig function results in the smallest reconstruction error. The horizontal line indicates the error achieved with full resolution.

WT-pyramid with $f(\omega_i) = \omega_i^2$ and (d) from the 3^{th} level of a WT-pyramid with $f(\omega_i) = \omega_i^3$.

Best results (Figure 19) on almost all levels were achieved with WT-contraction, if $f(\omega_i)$ was a modified logsig function 25 ($l = 9$,$t = 0.8$). Note that in figure 10(b)

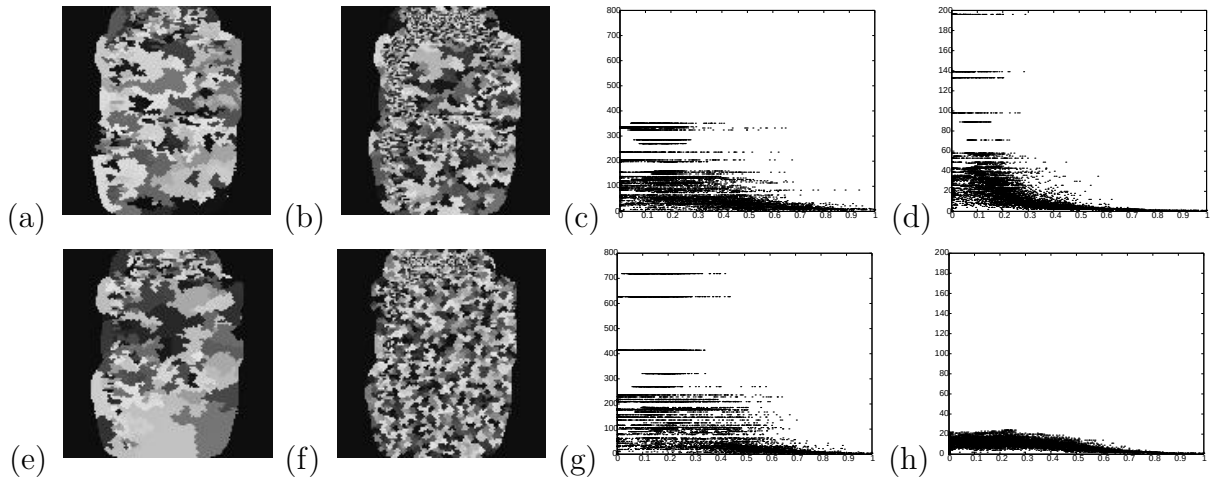


Figure 20: **Left part**: Receptive fields for WT contraction with $s = 2$ (upper row) and $s = 3$ (lower row). **Right part**: diagrams, where each point represents a pixel in the original image. The x-axis represents the weight, the y-axis the size of the receptive field in which it is contained.

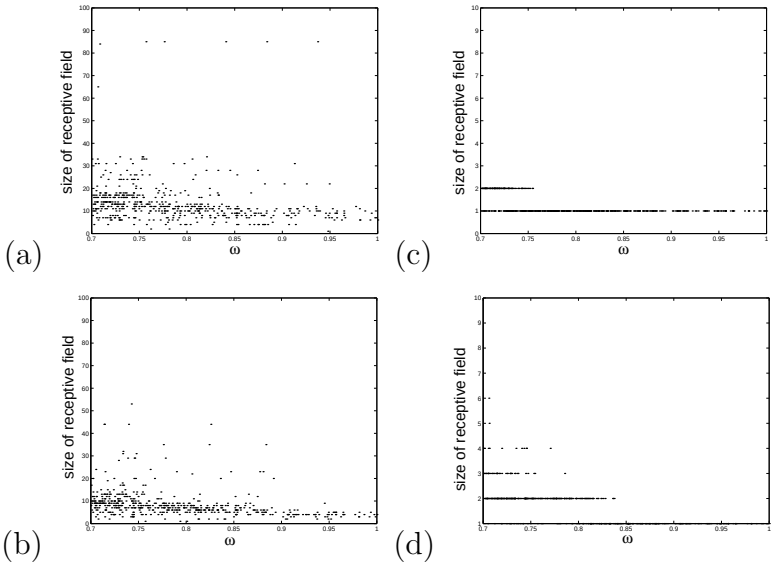


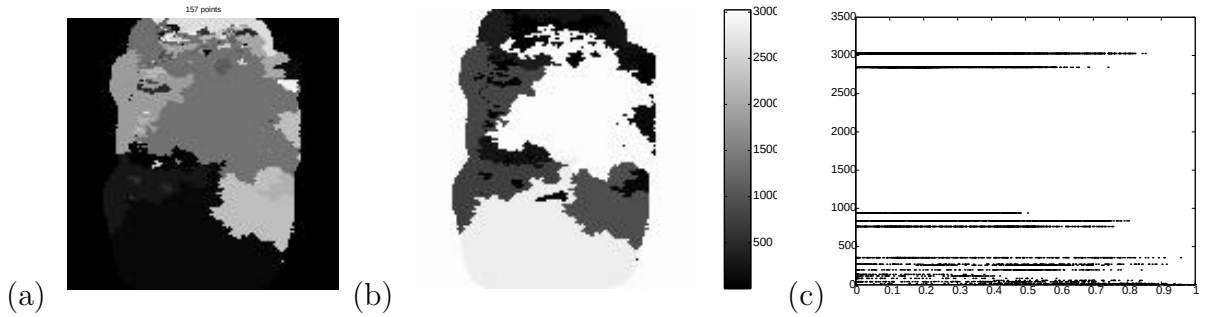
Figure 21: Phase diagrams: $s = 2$ with (a) 271 and (c) 1842 nodes; $s = 3$ with 283 (b) and 1559 (d) nodes.

$f(\omega_i) = logsig(\omega_i)$ provides smallest receptive fields for important regions and the steepest increase of receptive field size for decreasing weight. Figure 19 compares the performance of different functions $f(\omega_i)$ in WT-contraction. Another important observation is that optimal reconstruction results are not reached with highest reso-

lution but with a number of nodes ~ 3000 or $\sim 18\%$ of the original resolution.

2. **Reconstruction with high pyramid levels and the performance swap**
When the number of nodes is very low i.e. smaller than ~ 250 ($\sim 1.5\%$ of full resolution) DD-contraction outperforms WT-contraction independent of its parameter set or the used function $f(\omega_i)$ (in Figure 23 an arrow indicates the crossing of the performance curves. Note that in Fig. 19 and Fig. 23 the lines connecting the points do not reflect actual data, butconnect points of the same curve on the plot. Figure 22 helps to analyse the reasons. It represents the 9^{th} level of a pyramid constructed by *DD-contraction*. Note, that 5 extremly large receptive fields (i.e. > 700 nodes) cover almost the whole image. Regions with high *weight* are covered by these fields to a great extend. This is in contrary to WT-pyramids, where regions with wide differences in weight remain seperated. We explain this behaviour in the discussion section
Pyramids built by WT-contraction with $f(\omega_i) = \omega_i^s$ show the following behavior, when the number of nodes is decreased. Pyramids built with higher exponents s successively outperform or at least perform equally to pyramids built with lower exponents s.

Figure 24 shows a comparison of reconstruction errors achieved with different pyramids. The pyramids were built with WT-contraction and we set $s = 2, 3, 4$. We compared the MSE for reconstruction results of WT-contraction ($error_{WT}$) and results of a regular pyramid i.e. traditional downsampling ($error_{reg}$). In figure 24(a) one can see that the latter becomes unstable at resolutions below 500 pixels which is approximately 3% of the full resolution, whereas all irregular pyramids remain stable (i.e. $error_{reg}$ increases to values which are orders of magnitudes higher than $error_{WT}$). The plots show 3 different results of the regular pyramid to give an im-



(a) (b) (c)

Figure 22: Level 9 of a pyramid constructed by DD-contraction: (a) randomly colored receptive fields; (b) gray values according to the size of the receptive fields;(c) phase diagram, vertical axis: size of the receptive field, horizontal axis: weight.

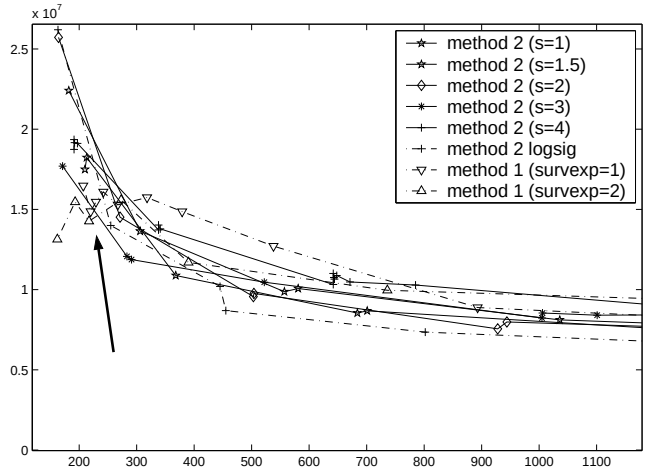


Figure 23: DD-contraction outperforms WT contraction when the number of nodes falls below ~ 250. The arrow in the plot indicates this point.

pression of their variations. Figure 24(b) shows a magnified part of 24(a). The arrows indicate the points where the performances of different s values cross.
On a close look we can observe the following property of pyramids built by DD-contraction: The method tends to enlarge receptive fields into regions despite their high weight but still some very small receptive fields remain in these areas. The phase diagram in figure 22(c) visualizes this behaviour. The x-axis indicates the weights of the pixels in the image, the y-axis indicates the size of the receptive field they

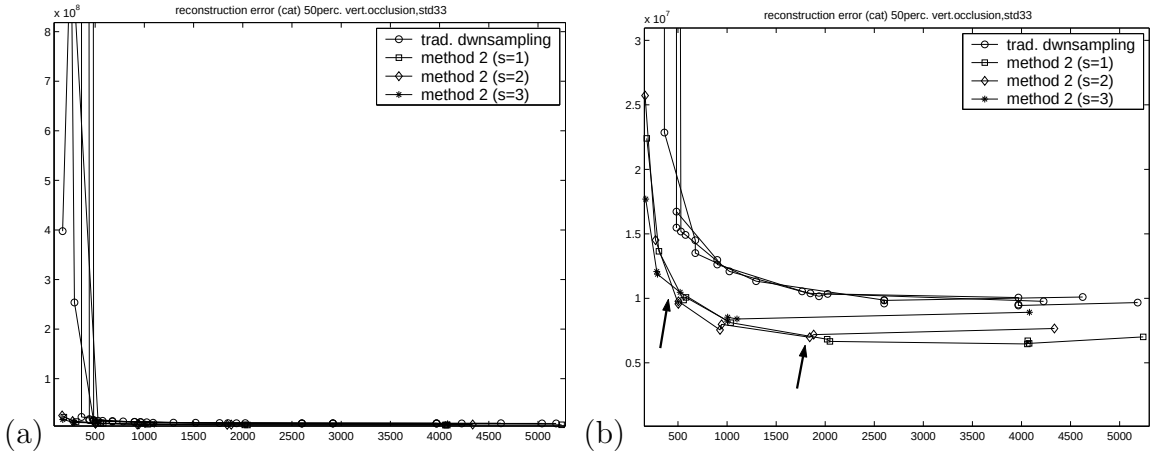


Figure 24: Comparison of the *mean squared error* of the reconstruction results after WT-contraction and regular pyramids, resp.

are part of. One can observe that a significant part of pixels with high weight lies in large receptive fields. Figure 22(a) shows randomly colored receptive fields while 22(b) indicates the size of the receptive fields by the gray value.

3. **Discussion of the results**

In this section we take a closer look at the receptive fields of the pyramids, to find the characteristics, that cause the differences in reconstruction accuracy and the performance swap.

For **WT-contraction** Figure 20 visualizes the dependencies between exponent s and the distribution of sizes of receptive fields. The figure is divided into two parts: (a,b,e,f) show randomly colored receptive fields, (c,d,g,h) show phase diagrams similar to figure 22 (c):

- Lower levels with 1842 and 1559 nodes resp., (b,d) and (f,h): the effect is inverted. Corresponding to figure 10 the lower exponent allowed faster contraction in regions with low weight (h). In figure 21(c,d) the receptive fields in regions with $\omega \geq 0.7$ are depicted. The effect of the fast contraction in low weight regions is the occurance of smaller receptive field sizes for more important regions.
- Higher levels with 271 and 283 nodes resp., (a,c) and (e,g): Here the higher exponent s causes a higher difference between the sizes of receptive fields in regions of higher and lower interest. The behavior is similar to DD-contraction. Because of extremely large receptive fields in less important regions there are enough nodes available to properly represent important regions. When comparing plot (c) and (g) one can notice that regions where $\omega \geq 0.8$ in plot (g) the sizes of the receptive fields are smaller than in (c). In figure 21 (a,b) this region is magnified.

Still all irregular pyramids outperform the regular downsampling process. (see also figures 24 and 23).

With **DD-contraction** we are not able to control the size of the receptive fields directly linked to the weights of the pixels. In section 2.3 we explained the effect of a distance function independent from the weights. It causes the spreading of large receptive fields into regions with high weight. The side effect of the occurrence of large receptive fields is that for a given number of nodes, in return there is a considerably larger amount of nodes representing very small regions. Figure 22 illustrates this behavior.

This seems to be a crucial feature when resolution becomes extremely low. Nevertheless we have to consider the experimental setup, where the right half of the picture is occluded completely and no noise does appear. Little information is lost by covering a part (approximately 2/3) of the important regions by large receptive fields. Small scattered receptive fields remain, and because neighbouring regions are highly correlated they contain enough information about the 'lost' regions with high weight.

If the number of nodes is above ~ 250 *WT-contraction* outperforms *DD-contraction*. During contraction the algorithm sets increasing upper bounds to the size of the receptive fields depending on the weights of the pixels lying within. This direct dependence achieves our goal to represent regions according to their importance best.

In practical application, where noise can appear in addition to occlusion we expect WT contraction to perform better even in levels with very low resolution. The reason lies in the stabilization against occluded or disturbed pixels in important areas provided by a 'proportionate' representation of this regions. The ability to suppress irrelevant information becomes obvious when comparing the reconstruction errors from full resolution images and the ones from images contracted by our algorithm. Figure 19 indicates the reconstruction error using full resolution with a horizontal line.

5 Conclusion

This report explores the possibilities to support appearance-based recognition and reconstruction by a top down strategy. This strategy was motivated by natural behavior of humans when they look at scenes, objects or faces. To realize a simulation of such a behaviour in the framework of robust PCA methods, irregular pyramids are applied to the images. Two algorithms to build these pyramids were introduced and discussed. In the last section experiments showed significant differences between regular downsampling and our approach regarding reconstruction accuracy.
Standard PCA is unstable against occlusion or outliers. The robust PCA approach [1, 5] improved and stabilized reconstruction results. In experiments with 50%-occlusion it decreased the error to $14\% - 28\%$ of its original value.
During our experiments applying irregular pyramids decreased the error of robust PCA to $\sim 53\%$ of the error achieved with regular downsampling at a resolution of ~ 3000 nodes ($\sim 18\%$ of full resolution). With less than 500 nodes ($\sim 3\%$ of full resolution) regular downsampling causes instability (Figure 24 and 19). The reconstruction error with irregular downsampling also increases but the gap grows considerably to $error_{irregular} \sim 2\%$ of $error_{regular}$.

A very interesting conlusion is that irregularly downsampled images can improve reconstruction results gained with full resolution images. This indicates a deteriorating influence on robust PCA if unimportant regions are represented to an improper extent.
At the end of this section we suggest further research on the topic. Future work on this topic could include:

1. study the connection between correlation and weights in an image region to gain information about texture and shape of the object.

2. robustify the training phase of robust PCA methods against varying background by eliminating regions of high variance, but extremely small correlation between pixels neighbouring each other. Textures with very small ornaments might cause troubles.

3. study different weight maps according to the task which has to be solved. We expect the optimal weight maps for recognition, reconstruction and distinction of objects to be different.

4. adjust weight maps to the number of used eigenvalues

5. compare structure (for example spatial variance) of input images with the pyramid structure to gain information about occlusion etc.

References

[1] Alěs Leonardis, Horst Bischof, Robust Recognition Using Eigenimages, *Computer Vision and Image Understanding 78, 99-118(2000)*

[2] Horst Bischof, Walter G. Kropatsch, Neural Networks versus Image Pyramids, *technical report PRIP-TR-7, Insitute for Automation, Pattern Recognition and Image Processing Group, University of Technology, Vienna*

[3] Horst Bischof, Pyramidal Neural Networks, *Lawrence Erlbaum Associates, 1995, New Jersey*

[4] Horst Bischof, Aleš Loenardis Robust Recognition of scaled eigenimages through a hierarchical approach, *Computer Vision and Pattern Recognition, pages 664–670, 1998*

[5] Aleš Leonardis, Horst Bischof, Roland Ebensberger, Robust Recognition Using Eigenimages *technical report PRIP-TR-47, Institute for Automation, Pattern Recognition and Image Processing Group, University of Technology, Vienna*

[6] Walter G. Kropatsch, Digitales Sehen mit Bildpyramiden *technical report PRIP-TR-1, Institute for Automation, Pattern Recognition and Image Processing Group, University of Technology, Vienna*

[7] Walter G. Kropatsch, Irregular Pyramids *technical report PRIP-TR-5, Institute for Automation, Pattern Recognition and Image Processing Group, University of Technology, Vienna*

[8] Walter G. Kropatsch, Aleš Leonardis, Horst Bischof, Hierarchical, adaptive and robust methods for image understanding, *Surveys on Mathematics for Industry, 1999, 9:1-47*

[9] Walter G. Kropatsch, Building irregular pyramids by dual graph contraction *IEEE-Proc. Vision, Image and Signal Processing, 142(6):366-374.1995* 2, 3, 26 14 14 2, 3, 16, 26 3, 10 3 3, 11 3

[10] Jean-Michel Jolion, Data driven decimation of graphs *Proc. of GbR'01, 3rd IAPR Int. Workshop on Graph based Representations, p.105-114* 11

[11] Etienne Bertin, Horst Bischof, Voronoi Pyramids controlled by Hopfield Neural Networks *technical report PRIP-TR-24, Institute for Automation, Pattern Recognition and Image Processing Group, University of Technology, Vienna*

[12] Roland Glantz, Walter G. Kropatsch, Guided relinking of Graph Pyramids *Joint IAPR International Workshops on SSPR'2000 and SPR'2000, volume 1876 of Lecture Notes in Computer Science, pages 367-376, Alicante, Spain, August 2000. Springer, Berlin Heidelberg, New York.*

[13] Peter F.M. Nacken Image segmentation by connectivity preserving relinking in hierarchical graph structures, *Pattern recognition Vol.28, No.6, pp.907-920, 1995*

[14] Lawrence W. Stark, Claudio M. Privitera, Top-Down and Bottom-Up Image Processing *Int. Conference on neural networks Volume: 4 , 1997 Page(s): 2294 -2299 vol.4*

[15] Dimitri A. Chernyak, Lawrence W. Stark, Top-Down Guided Eye Movements, *IEEE transactions on systems man and cybernetics-part B cybernetics, Vol31, No.4, August 2001*

[16] Claudio M. Privitera, Lawrence W. Stark, Algorithms for Defining Visual Regions-of-interest: Comparison with Eye Fixations, *IEEE transactions on Pattern Analysis and Machine Intelligence, Vol.22, No.9, september 2000*

[17] Peter Meer, Stochastic Image Pyramids, *Computer Vision, Graphics, and Image Processing Vol.45, No.3 1989, pp.269-294*

[18] S.A.Nene,S.K.Nayar and H.Murase *Columbia Object Image Library (COIL-20)*, technical Report CUCS-005-96, Columbia University, New York