Johann Wolfgang Goethe-Universität
Frankfurt am Main

Fachbereich Informatik

Project SEMACODE: A Scale-invariant Object Recognition System for Content-based Queries in Image Databases

R. Brause, B. Arlt, E. Tratar

{brause, arlt, tratar}@informatik.uni-frankfurt.de

INTERNER BERICHT 11/99

Fachbereich Informatik
Robert-Mayer-Straße 11-15
60054 Frankfurt am Main

Abstract

For the efficient management of large image databases, the automated characterization of images and the usage of that characterization for searching and ordering tasks is highly desirable. The purpose of the project SEMACODE is to combine the still unsolved problem of content-oriented characterization of images with scale-invariant object recognition and model-based compression methods.

To achieve this goal, existing techniques as well as new concepts related to pattern matching, image encoding, and image compression are examined. The resulting methods are integrated in a common framework with the aid of a content-oriented conception. For the application, an image database at the library of the university of Frankfurt/Main (StUB; about 60000 images), the required operations are developed. The search and query interfaces are defined in close cooperation with the StUB project "Digitized Colonial Picture Library".

This report describes the fundamentals and first results of the image encoding and object recognition algorithms developed within the scope of the project.

The project SEMACODE is supported by the *German Research Foundation* (*Deutsche Forschungs-gemeinschaft* DFG) within the strategic research initiative "V^3D^2" ("Distributed Processing and Exchange of Digital Documents").

Table of Contents

1 Scale-invariant Recognition of Objects in Images

The main purpose of the project SEMACODE is the investigation and development of the fundamental principles and methods needed for an autonomous system which is able to recognize objects in digital images. Potential applications of such a system are content-based retrieval and automated indexing of large multimedia databases or digital libraries. For the example of the image database (about 60000 images) of the former "German Colonial Institution" (*Deutsche Kolonialgesellschaft*) at the university library of Frankfurt/Main (*Stadt- und Universitätsbibliothek Frankfurt/Main* StUB), the achieved results should be demonstrated. Furthermore, the interfaces of the developed search and query operations should be adapted to ease their integration into a standard database environment.

This report covers the recent theoretical and experimental research done within the scope of the project and concentrates on the efficient encoding of images for object recognition purposes. It also describes the principles and first results of the developed recognition algorithms.

1.1 Image Query System

The object recognition system will be integrated into an image query system which provides the necessary procedures and interfaces for searching digital images in multimedia databases or the internet. The image query system is principally based on the query-by-example approach: The user submits text search criteria and example images. The retrieved results are generated by the databases for images, text, and meta data, and the query mechanism itself. Thus, the whole system can be organized in a client-server structure as shown in Figure 1.

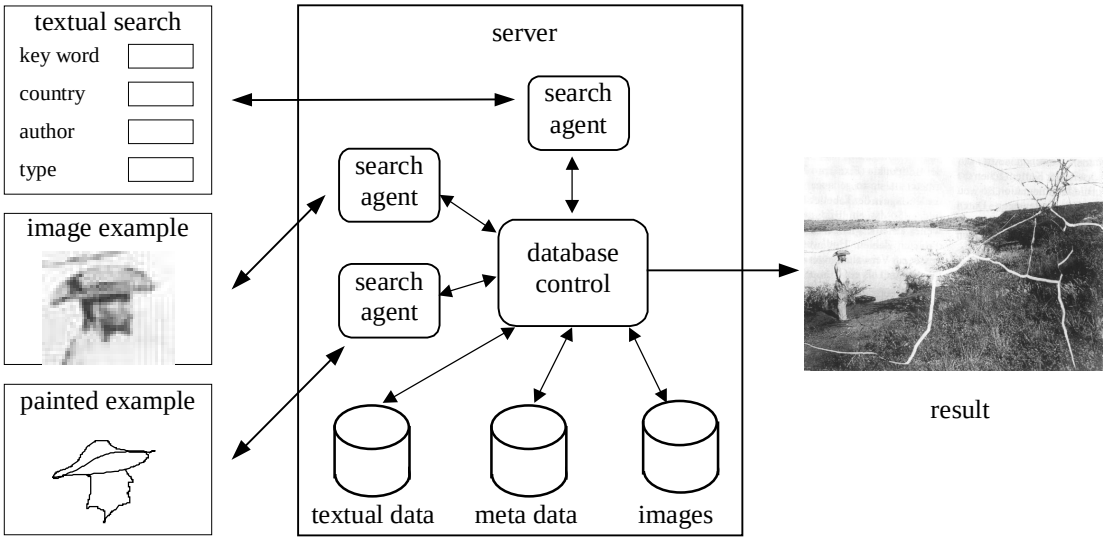


Figure 1: Data and control flow of the image query system.

The design of the system has to meet the following requirements:

- Intuitive and efficient handling of the tools for entering the search criteria; as an internet application, the tools should be used together with standard browsers;
- Fast and correct image query algorithms;
- Suitable representation of the retrieved search results.

The input of key words allows a conventional search for images using standard query methods of the database server. Thus, a previous classification of the images is required. However, the user should be able to search for objects or images which are possibly unknown to the system.

This can be accomplished by a technique called *query-by-example* (as used by other image query systems, see e.g. *QBIC* [QBC95]), i.e. the user may paint an object with a graphical editor or select it

from a given image. The resulting image is an example for the requested object. The system now identifies all images that are similar to the example or contain similar objects. This is done by starting several server processes (*search agents*) which perform the object recognition task. To obtain a reasonable performance the search agents operate on the encoded image data (*meta data*) instead of the images themselves. After completing the recognition task the systems returns the resulting images as preview versions in reduced size (*thumbnails*) ordered according to a suitable similarity measure. The user may now accept the results or select one of the images as an example to start another query.

The image database is expected to be globally accessible via internet. The query procedures were realized as Java-applets or CGI-scripts running at the corresponding WWW-server. Thus, global access of the query and representation operations via internet is possible. Figure 2 shows the interaction layers of the image query system, the web server, and the database server.

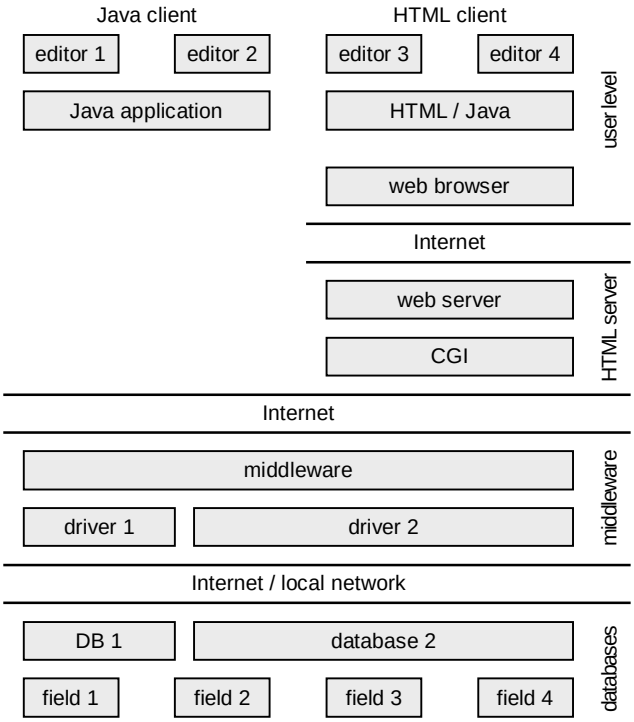


Figure 2: Layer structure of the distributed image query system.

A demonstration version of the system is accessible at „http://seco.asa.cs.uni-frankfurt.de/seco.html“. Currently, only simple search criteria such as textual information and intensity distribution of pixel data are implemented. Future improvements include the integration of new developed object recognition methods which are described in the following of this paper.

1.2 Object Recognition System

The main subject of the project SEMACODE is the application of semantic image encoding. Here the necessary compression and semantic description of images can be combined by using the concepts of “image primitives” and “image elements”. This kind of encoding is suitable for both storage and object recognition tasks and therefore allows an efficient processing of images.

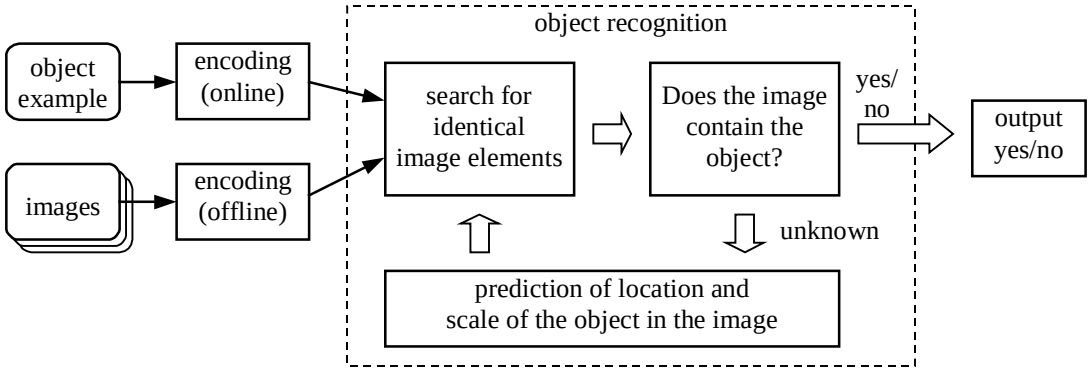


Figure 3: Structure of the object recognition system.

Figure 3 shows the structure of the object recognition system. The images are already encoded offline and stored in the (meta) database. At the beginning of the query only the requested object has to be encoded online. The system searches for identical image elements and predicts the possible location and size (scale) of the object in the image. If such elements exist and the prediction is correct the system "recognizes" the object. Otherwise a new prediction is calculated or (if the system cannot predict the object's location and/or scale) the output "object not recognized" is given.

According to the difficulties of recognizing objects in images described in section 1.3 the system has to meet several requirements:

- *Scale invariance*
  Since the size of a requested object is *a priori* unknown the system must be able to recognize objects regardless of their scale.
- *Translation invariance*
  Objects have to be recognized regardless of their location in the actual image.
- *Robustness against individual parameters of images*
  Individual parameters of images such as illumination or contrast must not affect the system's functionality.
- *Correctness and reliability*
  The image query system has to be correct and reliable. There are two classes of possible errors: The incorrect recognition of objects which *are not* included in a given image, and the failed recognition of objects which *are* included.
- *Efficiency*
  The system has to be efficient and fast.
- *Further invariance*
  If possible, the system should also recognize objects which are moderately rotated, mirror-inverted, distorted, or partially occluded by other objects or image borders.

1.3 The Problem of Recognizing Objects in Images

An object is a real-life item (e.g. a chair, a house), an abstract entity, or a symbol (e.g. a character or letter). Objects may have different instances (e.g. block houses or skyscrapers, Latin or Chinese letters) and can be composed of other objects (a house is "composed" of walls, a roof, windows, doors,…).

Here, an object is assumed to be given by an entity which is consistent according to the actual semantic context (*semantic entity*). Each object possesses certain features or *attributes* describing the object with more or less accuracy. In their own semantic context attributes can be objects themselves.

The automated recognition of objects is a very hard image processing task. In the following sections a brief summary of the various problems which make recognizing objects in images difficult is given.

1.3.1 Ambiguity

The goal of an object recognition system is the identification of real-life objects by their two-dimensional representations or images respectively. Paradoxically, this leads to the conclusion that the unambiguous recognition of objects without additional information and explicit knowledge is impossible – recognizing objects in images is an *ill-posed problem*. In Figure 4 this is illustrated by a simple example.

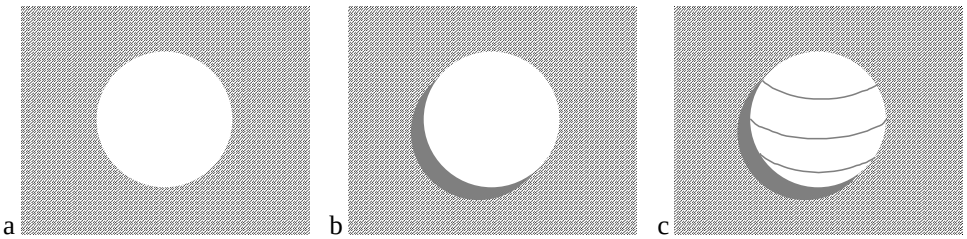


Figure 4: Using additional image features and explicit information to remove ambiguity of an object representation.

It is not possible to infer unambiguously if Figure 4a represents a sphere, a circle, a rectangle with a hole, or something different. Figure 4b contains an additional image feature: The shadow suggests a three-dimensional appearance of the object. However, Figure 4b might also represent a gray circle partially occluded by a white circle. The lines in Figure 4c emphasize the impression of the object's plasticity and with high probability one recognizes a three-dimensional sphere although other interpretations of the image might exist.

Thus, *implicit* information contained in images in general is not sufficient to identify objects unambiguously. Obviously, a mechanism exists which enables human beings to assign different probabilities to various interpretations of an image: The object will be recognized if its representation is the most probable one. Here, explicit knowledge about real world examples is used which cannot be derived from the image alone (e.g. knowledge about the shadow of a three-dimensional sphere as seen in a two-dimensional image). The mathematical counterpart is given by *regularization theory* (see [PT85]) where additional rules and/or restrictions are used to transform ambiguous and incompletely formulated problems into well-posed problems.

1.3.2 Scale and Position of Objects

The system has to recognize objects in a given images regardless of their location and size. This turns out to be a very difficult tasks, especially since it is unknown if an image contains a special object.

Besides the computational complexity caused by the search for objects in a large scale of possible sizes and image locations, a systematical problem has to be considered: An unambiguous correspondence between images showing the same object at different size or scale does not exist in general.

For example, decreasing the size may remove some details of the image or merge them into new details which do not correspond to any elements of the original image. Furthermore, small shifts (translations) in the range of a few pixels may result in a different encoding of an object. Thus, the recognition system has to take care of these effects.

1.3.3 Rotation and Deformation

Another requirement could be that the system is able to recognize rotated, mirror-inverted, and distorted instances of an object as well. However, in some cases this is not an appropriate constraint: Human beings do distinguish between the letters "M" and "W" although both are rotated versions of each other (ignoring the various typefaces). On the other hand, a pebble should be recognized regardless of its orientation. Similar examples also can be found for reflected or distorted objects.

The problem gets even more complicated if the orientation of the image itself is unknown. This is a typical situation in multimedia databases where images are stored in different orientations to fit a given image size and are erroneously scanned upside down or mirror-inverted.

1.3.4 Individual Parameters of Images and Objects

Images may differ in various parameters such as intensity, contrast, and sharpness. Other parameters are disturbances due to noise or scanning of already corrupted images. Furthermore, the represented objects themselves may differ in their appearance because of varying illumination. Although it is possible to compensate variations of overall image parameters the problem of object illumination is rather difficult and requires non-trivial solutions.

1.3.5 Occlusion

Another difficulty arises from the typical occlusion of objects in two-dimensional images. A 3D-object may occlude parts of itself (e.g. as a result from its three-dimensional nature) or other objects. Often parts of objects are occluded by image borders as well.

In the presence of occlusion the identification of objects often becomes ambiguous or impossible because salient object parts are invisible. In addition, the recognition of the *occluding* object is affected e.g. by erroneously interpreting parts of an occluded object as parts of the current object.

The problem of self-occlusion is related to the theory of *view-based recognition*. Here the main subject of investigation is the question if objects could be recognized when looking at them from different viewing angles. There are several indications that this is indeed possible, even if only a few different views are used [BRE93] [LVH94].

1.3.6 Elements of Images

Until now it was assumed that parts of a given image (*image elements*) could be identified with objects or parts of objects. However, the nature or properties of image elements is still unclear.

For example, connected image regions (*segments*) sharing similar features may assumed to be image elements; therefore an image can be decomposed into non-overlapping segments. The borders of segments are often determined by edges or lines (*edge-oriented segmentation*). Another technique results from merging small parts of an image if they contain similar patterns or textures (*texture-based segmentation*). However, for a successful segmentation the choice of the operators detecting the required image features is crucial. Here the use of regularization techniques (e.g. relaxation of broken segment borders) is mandatory. Typical problems of image segmentation are shown in Figure 5.

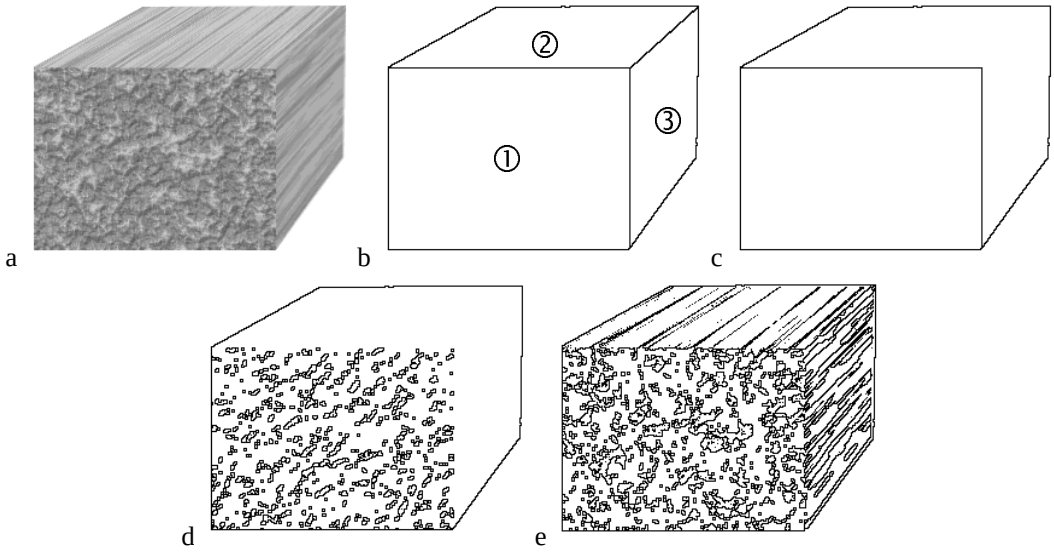


Figure 5: Example of segmentation problems: The original image (a) and three segments (b); typical results of texture-based segmentation (c) and edge-oriented segmentation with different thresholds (d, e).

Perfect segmentation of the cube in Figure 5a would result in three regions (the visible sides of the cube as shown in Figure 5b). In most cases texture-based segmentation produces only two regions: The sides 2 and 3 have nearly identical textures and therefore are merged into one segment (Figure 5c). In contrary, edge-based segmentation techniques determine the borders of segments while searching for significant changes in local pixel intensity and/or color. The determination of segment borders usually depends on the strength of the intensity (or color) gradient and is therefore threshold-driven. For images with high-contrast textures, this often leads to incorrect segmentation as shown in Figure 5d+e.

A different approach is the identification of image elements with groups of image features that do not necessarily need to form connected regions (see Figure 6); considering occlusion effects this is of particular interest. However, in general it is not obvious which image features should be grouped to the same image elements or what "typical" image features are.

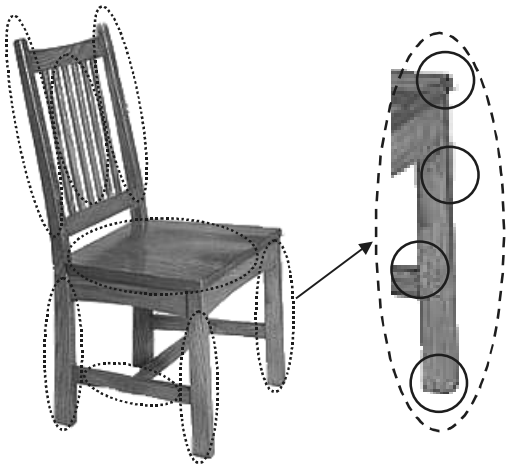


Figure 6:
The representation of a chair and its possible image elements (marked by dotted ellipses). Every image element results from the grouping of not necessarily connected image features (circles) as shown in the magnification.

1.3.7 Image Features and Image Primitives

Independent of the methods used to extract objects or image elements from images an encoding scheme is needed which mediates between the pixel data and the abstract representation level of objects. This scheme is given by the image features already considered above. The intention of (*low-level*) *image features* is to locally characterize small areas of a given image; this could be lines, edges, textures, or something similar.

An approach widely used in image processing tasks is the representation of small image parts by mixtures of image features (*image primitives*). Image primitives may be applied to edge or texture segregation as well. Again, the choice of suitable image primitives is a non-trivial problem.

1.3.8 Object Recognition and Computational Complexity

A typical digital image is about a few Kilobytes to some Megabytes in size. Considering the difficulties mentioned above and the expected computational complexity of the algorithms, the amount of underlying image data has to be reduced dramatically or the automated recognition of objects in images can not be done efficiently in real-time.

Here, the focus is set on the encoding of images by only a few salient primitives, which provides a more compact representation than the original pixel data. This is a critical task: In general, to achieve a high compression ratio a *lossy encoding scheme* is mandatory, i.e. redundant and unimportant information is removed from the image data by using the image primitives and can not be used by subsequent processing stages. Obviously, special care has to be taken of the choice of the image primitives.

Furthermore, one has to pay attention to the computational complexity of high-level processing tasks as well: Image database queries have to be executed correctly and as fast as possible.

2 Encoding Scaled Images

This chapter describes the work of the project SEMACODE to derive an efficient image encoding strategy which is needed to develop a successful object recognition system. Following a brief introduction of the fundamental concepts, recent research results are given in detail.

2.1 Image Encoding Using Image Primitives

Many approaches concerning the recognition of objects in digital images are based on the idea of representing an image by suitable image features or image primitives. Because of the following reasons, an actual search or recognition task does not use the pixel data but the primitives:

- Image primitives provide a (low-level) content-based characterization of an image. A primitive usually has a straightforward semantic equivalent such as *line*, *edge* or *texture*. Thus, the content-based analysis done by subsequent processing stages (e.g. segmentation or grouping) will be simplified.
- Ideally, image primitives represent "salient" information, i.e. the information needed for the actual image processing task; noise or redundant information will be removed from the data.
- Image primitives provide a compact representation (encoding) of images; the amount of the image data will be reduced (compression). This is a mandatory precondition for many image processing tasks.

Thus, the encoding of images by image primitives is an important preprocessing step to simplify and speed up subsequent stages. The choice of the set of primitives substantially contributes to the efficiency of this encoding.

2.1.1 Existing Methods

There exist many methods to extract simple image elements such as lines or edges from images, which, for example, can be used to track contours of objects. Most important are those methods where the associated feature detectors or *filters* (such as Scale-Space filters [LIN94] and regularization operators [PT85]) are robust against the influence of noise.

Another class of image primitives is given by *Gabor filters* [GAB46]. Gabor filters provide a local representation of images in the frequency domain and have been successfully used in texture segmentation tasks [GRE96] [HPB98] [PZ89]. Standard edge detectors often fail to decompose an image into separated regions (segments) because the region borders consist of different adjacent textures instead of large intensity gradients.

Besides there are adaptive techniques extracting the associated image primitives according to statistical properties of the image data. A widely used method is *Principal Component Analysis* (PCA; see for example [BRA95]) based on *transform coding* principles. Here an image is subdivided into small non-overlapping "patches" (*subimages*) that are about 8×8 to 16×16 pixels in size. Each subimage is represented by a linear mixture of a set of orthonormal image primitives. The primitives correspond to the eigenvectors of the covariance matrix of the subimages and thus are sometimes called *eigenimages*. It can be proved that the representation of an image reconstructed by the first eigenvectors with the highest eigenvalues is optimal in the mean square error sense.

An important property of the PCA is that the mixture components representing the influence of the associated eigenvectors on each subimage are decorrelated. Decorrelation is a necessary but not sufficient condition for statistical independence, i.e. statistical independent components have to be decorrelated. However, decorrelated components do not have to be statistical independent, except if their probability density functions (*PDF*) are Gaussians.

The goal of *Independent Component Analysis* (ICA; see for example [COM94]) is the derivation of components which are as statistical independent as possible. In general, the resulting primitives are not orthonormal, and the associated filters derived from "natural" image data are very similar to Gabor filters or the oriented receptive fields of the visual cortex, respectively [BS96] [OF96].

Independent components of natural images have two advantageous properties: Ideally, they are not redundant and sparsely coded [OLS96], i.e. only a few components are active at the same time. From the

viewpoint of information theory, both properties allow an efficient compression of the image data. Furthermore, independent components are assumed to represent *causal independent* image elements and thus provide the "most salient" information of an image.

2.2 Adaptive Image Encoding

This section gives a brief introduction to the mathematical fundamentals and properties of the PCA and ICA based on a technique known as *transform coding.*

2.2.1 Transform Coding

Transform coding of image data (see for example [HW71]) is a general encoding framework where images are decomposed into small non-overlapping subimages (about 8×8 to 16×16 pixel in size). Writing its pixel rows or columns in a sequential order, each subimage corresponds to an *image vector* $\mathbf{x} = (x_1,\ldots,x_n)^T$ (see Figure 7). The coefficients x_i of $\mathbf{x}$ represent the value (intensity, color) of the associated pixels. Here, the average $\langle \mathbf{x} \rangle$ of the image vectors $\mathbf{x}$ is assumed to be zero, otherwise any non-zero average has to be subtracted from the whole ensemble.

The image vectors $\mathbf{x}$ are transformed into component vectors or *jets* $\xi = (\xi_1,\ldots,\xi_n)^T$ by a matrix $\mathbf{A}$: $\xi = \mathbf{A}\cdot\mathbf{x}$. The rows of $\mathbf{A}$ are often called *analysis filters*. Ideally, only a small number of components $\xi_1,\ldots,\xi_m$, $m < n$, is sufficient to represent the image vectors (data compression by sparse coding) and to process them in subsequent stages (object recognition, storage, transmission, or others) [1].

In general it is possible to reconstruct the vectors $\mathbf{x}$ by the transform $\mathbf{x} \approx \mathbf{x'} = \mathbf{A}^\dagger\cdot\xi$ where the matrix $\mathbf{A}^\dagger$ is not necessary identical to the inverse matrix of $\mathbf{A}$. The columns of $\mathbf{A}^\dagger$ are called *synthesis filters* (or *base images* if they constitute a basis of the reconstructed image vectors $\mathbf{x'}$). Figure 8 shows the processing scheme of encoding images based on the transform coding framework.

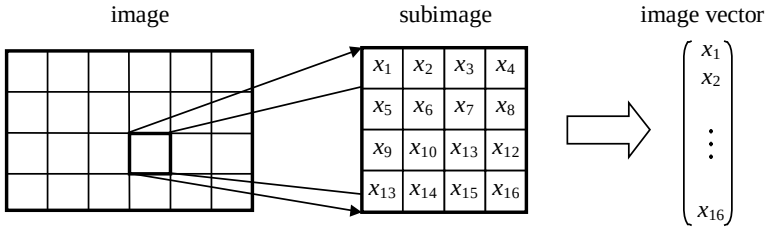


Figure 7: Image, subimage, and image vector.

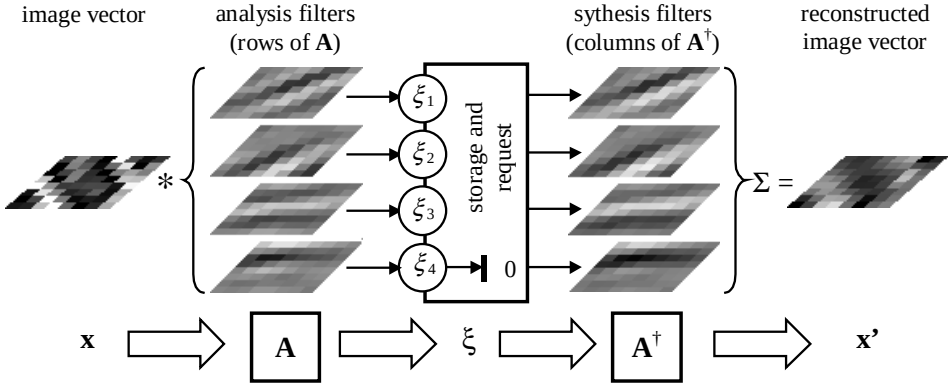


Figure 8: Processing scheme of transform coding. The image vector $\mathbf{x}$ is transformed into the jet ξ by a matrix $\mathbf{A}$. The actual processing of the image data (e.g. object recognition, storage, transmission) is executed on the first m components $\xi_1,\ldots,\xi_m$ of ξ. In general, the image vector can be reconstructed from these components by the transform $\mathbf{A}^\dagger$.

[1] The same technique is used in conjunction with a Discrete Cosine Transform to compress JPEG images.

2.2.2 Principal Component Analysis (PCA)

To calculate the PCA encoding of the image vectors $\mathbf{x}$ according to the above scheme, their covariance matrix $\mathbf{C}$ has to be determined first. Since $\langle \mathbf{x} \rangle = \mathbf{0}$ the covariance matrix results from the expected outer product of the image vectors: $\mathbf{C} = \langle (\mathbf{x}-\langle \mathbf{x} \rangle)\cdot(\mathbf{x}-\langle \mathbf{x} \rangle)^T \rangle = \langle \mathbf{x}\cdot\mathbf{x}^T \rangle$. The normalized eigenvectors $\mathbf{e}_i$ of $\mathbf{C}$ are called the *PCA primitives* or *eigenimages* of the vectors $\mathbf{x}$. Because of the symmetry of $\mathbf{C}$ they constitute an orthonormal basis; their eigenvalues λ_i are real and non-negative.

The first $m \le n$ eigenvectors $\mathbf{e}_i$ ordered according to their descending eigenvalues λ_i and written as column vectors of a matrix $\mathbf{K}_m$ allow the PCA transformation

$$\mathbf{u} = \mathbf{K}_m^T \cdot \mathbf{x} \tag{1}$$

The transformed image vectors $\mathbf{u} = (u_1,\ldots,u_m)^T$ are called *PCA jets*.

Since the columns of $\mathbf{K}_m$ are orthogonal the inverse is given by $(\mathbf{K}_m)^{-1} = \mathbf{K}_m^T$ if $m = n$. However, $\mathbf{K}_m$ can not be inverted if $m < n$. In this case $\mathbf{K}_m^T$ represents the *pseudo inverse* of $\mathbf{K}_m$. The reconstruction $\mathbf{x'} = \mathbf{K}_m \cdot \mathbf{u} \approx \mathbf{x}$ of an image vector is optimal in the sense of the *mean square error* (MSE):

$$\mathrm{MSE}(\mathbf{x},\mathbf{x'}) = \left\langle (\mathbf{x} - \mathbf{x'})^2 \right\rangle \tag{2}$$

Comparing to the notation in section 2.2.1 the correspondences $\xi \equiv \mathbf{u}$, $\mathbf{A} \equiv \mathbf{K}_m^T$, and $\mathbf{A}^\dagger \equiv \mathbf{K}_m$ are obtained.

The components u_i of the PCA jets $\mathbf{u} = (u_1,\ldots,u_m)^T$ are centered since $\langle \mathbf{u} \rangle = \mathbf{K}_m^T \cdot \langle \mathbf{x} \rangle = \mathbf{0}$. Furthermore, they are decorrelated:

$$\langle \mathbf{u}\cdot\mathbf{u}^T \rangle = \left\langle \mathbf{K}_m^T \cdot \mathbf{x} \cdot \mathbf{x}^T \cdot \mathbf{K}_m \right\rangle = \mathbf{K}_m^T \cdot \langle \mathbf{x}\cdot\mathbf{x}^T \rangle \cdot \mathbf{K}_m = \mathbf{K}_m^T \cdot \mathbf{C} \cdot \mathbf{K}_m = \mathbf{K}_m^T \cdot \Lambda \cdot \mathbf{K}_m = \Lambda \tag{3}$$

The covariance matrix $\Lambda = diag(\lambda_1,\ldots,\lambda_n)$ of the u_i is a diagonal matrix and the variances $var(u_i)$ are given by the eigenvalues λ_i. Thus, the components u_i (the *principal components*) are ordered according to their descending variances.

2.2.3 Independent Component Analysis (ICA)

The ICA is based on a *mixture model*: The image vectors $\mathbf{x}$ are assumed to be generated by a linear mixture of n unknown, statistically independent, and centered *sources* $\mathbf{s} = (s_1,\ldots,s_n)^T$, where $\langle \mathbf{s} \rangle = \mathbf{0}$ and $p(\mathbf{s}) = p(s_1) \cdot p(s_2) \cdot \ldots \cdot p(s_n)$. Here, $p(.)$ represents the probability density function (PDF). The *mixture matrix* $\mathbf{M}$ is unknown as well but has to be non-singular (i.e. invertible):

$$\mathbf{x} = \mathbf{M} \cdot \mathbf{s} \tag{4}$$

The goal is to determine the inverse matrix $\mathbf{M}^{-1}$ and to reconstruct the sources $\mathbf{s}$ from the given image vectors: $\mathbf{M}^{-1} \cdot \mathbf{x} = \mathbf{M}^{-1} \cdot \mathbf{M} \cdot \mathbf{s} = \mathbf{s}$. However, the ICA is an ill-posed problem: If at most one source s_i is a Gaussian, the inverse mixture process can only be identified up to an unknown diagonal matrix $\mathbf{D}$ and a permutation matrix $\mathbf{P}$ (see for example [AB98a]):

$$\mathbf{y} = \mathbf{B} \cdot \mathbf{x} = \mathbf{B} \cdot \mathbf{M} \cdot \mathbf{s} = \mathbf{D} \cdot \mathbf{P} \cdot \mathbf{s} \tag{5}$$

The matrix $\mathbf{B}$ and the vectors $\mathbf{y}$ are called the *demixing matrix* and the *reconstructed sources* respectively. Since it is impossible to identify $\mathbf{D}$ and $\mathbf{P}$ unambiguously, the sources are assumed to have unit variances $var(s_i) \equiv 1$ for all i, and $\mathbf{P}$ is set to the identity matrix: $\mathbf{P} \equiv \mathbf{I}$. Thus the reconstructed sources $\mathbf{y}$ equal the original sources $\mathbf{s}$ up to their signs. According to section 2.2.1 the following correspondences hold: $\xi \equiv \mathbf{y}$, $\mathbf{A} \equiv \mathbf{B}$, and $\mathbf{A}^\dagger \equiv \mathbf{B}^{-1}$.

A simplification of the calculation of $\mathbf{B}$ is obtained if the image vectors $\mathbf{x}$ are *whitened*, i.e. if they are transformed by a *whitening matrix* $\mathbf{Z}$. The whitened image vectors $\mathbf{v} = \mathbf{Z} \cdot \mathbf{x}$ satisfy the equation

$$\langle \mathbf{v}\cdot\mathbf{v}^T \rangle = \left\langle \mathbf{Z}^T \cdot \mathbf{x} \cdot \mathbf{x}^T \cdot \mathbf{Z} \right\rangle = \mathbf{Z}^T \cdot \langle \mathbf{x}\cdot\mathbf{x}^T \rangle \cdot \mathbf{Z} = \mathbf{Z}^T \cdot \mathbf{C} \cdot \mathbf{Z} = \mathbf{I} \tag{6}$$

The whitening matrix $\mathbf{Z}$ can be derived[2] from the PCA transform of the vectors $\mathbf{x}$: $\mathbf{Z} \equiv \mathbf{W}_{pca} = \Lambda^{-½} \cdot \mathbf{K}_n^T$ with $\Lambda^{-½} = diag(\lambda_1^{-½},\ldots,\lambda_n^{-½})$ (see equation (3)). Thus, the determination of $\mathbf{B}$ reduces to the determination of an orthogonal matrix $\mathbf{W}_{ica}$ with $\mathbf{B} = \mathbf{W}_{ica} \cdot \mathbf{W}_{pca}$; since the reconstructed sources y_i are decorrelated (because of their statistical independence), centered (i.e. $\langle \mathbf{y} \rangle = \langle \mathbf{s} \rangle = \mathbf{0}$), and have unit variances $var(y_i) = 1$ for all i, the following equation holds:

$$\begin{aligned} \mathbf{I} &= \langle \mathbf{y} \cdot \mathbf{y}^T \rangle = \mathbf{B} \cdot \langle \mathbf{x} \cdot \mathbf{x}^T \rangle \cdot \mathbf{B}^T = \mathbf{W}_{ica} \cdot \mathbf{W}_{pca} \cdot \langle \mathbf{x} \cdot \mathbf{x}^T \rangle \cdot \mathbf{W}_{pca}^T \cdot \mathbf{W}_{ica}^T \\ &= \mathbf{W}_{ica} \cdot \langle \mathbf{v} \cdot \mathbf{v}^T \rangle \cdot \mathbf{W}_{ica}^T = \mathbf{W}_{ica} \cdot \mathbf{I} \cdot \mathbf{W}_{ica}^T = \mathbf{W}_{ica} \cdot \mathbf{W}_{ica}^T \\ &\Rightarrow \quad \mathbf{W}_{ica}^T = (\mathbf{W}_{ica})^{-1} \quad \Rightarrow \quad \mathbf{W}_{ica} \text{ is orthogonal.} \end{aligned} \tag{7}$$

Methods to calculate $\mathbf{W}_{ica}$ can be found for example in [COM94], [HO97], or [ARL97]. Figure 9 shows the block structure of the different processing stages of the ICA.

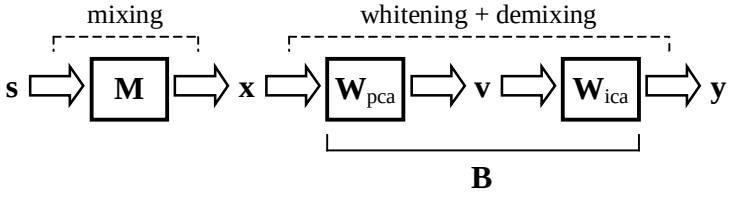


Figure 9: Processing stages of ICA.

2.3 New Methods for Encoding Images

A combination of PCA and ICA called *Principal Independent Component Analysis* (PICA) [AB98a] [AB98b] [AB98c] results from the calculation of the m independent components from the first m *principal components* (the m components of the PCA with highest variance). Here, the mean square error of the reconstruction is equal to the one of the PCA, but the derived *principal independent components* (PIC) are "more independent" than the principal components. Thus, PICs provide better encoding properties.

Because of the following advantages, especially PCA and PICA techniques have been investigated in the scope of the project SEMACODE:

- In contrary to other techniques such as *Discrete Fourier*, *Cosine*, or *Gabor Transform* (DFT, DCT, DGT), PCA and PICA implicitly utilize the statistical properties of the image data; therefore an efficient encoding scheme is derived.
- PCA and PICA are optimal in the mean square error sense because the primitives are adapted to the given image data.

A drawback of PCA and PICA is that (in theory) the primitives have to be calculated separately for each image. Nevertheless, in case of the PCA this problem can be overcome by utilizing a simple and reliable heuristic.

The investigation of PCA and PICA primitives has been expanded to overlapping subimages as well. A special technique where a two-dimensional Gaussian weighting function is applied to the subimages prior to the calculation of the primitives leads to the *Gaussian-windowed PCA* or *PICA* (GPCA, GPICA) respectively. The idea of applying the weighting function is to accentuate the center and to attenuate the borders of subimages which is a well-known property of oriented receptive fields. Here, the resulting image primitives turned out to be independent of the size of the analyzed images.

[2] Of course, other methods to derive the whitening matrix $\mathbf{Z}$ exist. For example, Bell and Sejnowski use a matrix $\mathbf{Z}$ satisfying the condition $\mathbf{Z}^T \cdot \mathbf{Z} = \mathbf{C}^{-1}$ [BS96].

2.3.1 Independent Components Analysis of the PCA Subspace (PICA)

The number of the independent sources s_i is equal to the number n of the components of an image vector $\mathbf{x}$. The question is whether all sources or just a few "salient" are needed to accomplish a given image processing task (for example the recognition of objects).

The measure of the "saliency" of a PCA component u_i is implicitly given by its variance $var(u_i)$: The variance corresponds to the amount of the associated eigenvector $\mathbf{e}_i$ in the mixtures representing the image vectors. In case of the ICA no such measure is given: The variances of the sources are identical (= 1), and even the average content of information (*marginal entropy*) or the *virtual variance* [3] do not provide a significant indication for the saliency of an ICA component [AB98a].

However, the independent sources can be calculated for the subspace U_m, which is spanned by the eigenvectors $\mathbf{e}_1,\ldots,\mathbf{e}_m$ ($m < n$) of the covariance matrix $\mathbf{C}$. This is done by using the whitening matrix $\mathbf{W}_{pca,m} = \Lambda_m^{-½} \cdot \mathbf{K}_m^T$ with $\Lambda_m^{-½} = diag(\lambda_1^{-½},\ldots,\lambda_m^{-½})$ instead of the matrix $\mathbf{W}_{pca} = \Lambda^{-½} \cdot \mathbf{K}_n^T$. U_m constitutes the m-dimensional subspace with maximum variance (*PCA subspace*) of the vector space spanned by the image vectors $\mathbf{x}$. Since $\mathbf{W}_{ica}$ is orthogonal, the rows $\mathbf{b}_i$ of the ICA matrix $\mathbf{B} = \mathbf{W}_{ica} \cdot \mathbf{W}_{pca}$ span U_m as well. Thus, only the "most salient" sources y_i with highest influence, the *principal independent components* (PIC), are extracted; the mean square error of the reconstruction is identical to the one of the corresponding PCA transform. Further details can be found in [AB98a], [AB98b], and [AB98c].

2.3.2 Image Encoding Using Gaussian-weighted Subimages

The different encoding techniques described in section 2.2 are based on the analysis of small rectangular patches (subimages) of a given image. Thus, the extracted image primitives have the same rectangular shape as the subimages. Since primitives are assumed to represent certain image features, their rectangular shape appears to be somehow artificial and unnatural. Furthermore, rectangular primitives lead to undesirable reconstruction effects of the whole image as shown by theoretical and experimental results (*tiling*; see for example [GW93] and Figure 16b).

Tiling can be avoided by reasonable choice of the subimages. The column vectors of an inverse or pseudo inverse transform $\mathbf{A}^\dagger$ are considered to be image features or primitives whereas the row vectors of the transform $\mathbf{A}$ represent their corresponding detectors or filters. The coefficients ξ_k ($1 \le k \le m$) of the jets ξ specify the influence of the k^{th} primitive on the current subimage.

In biological visual systems, the receptive fields are known to extract certain image features as well. Here, points at the center of a receptive field have a higher influence on the output of the associated cell than points at the borders. The shape of the corresponding weighting function often resembles a two-dimensional Gaussian (see Figure 11b below) as used by the Gabor transform [GAB46]. In contrary, techniques based on standard transform coding apply a uniform weighting function (see Figure 11a).

The generation of the subimages can be visualized by sliding a *window F* over a given image: Pixels inside the window belong to a new subimage (*sliding window technique* [KK92] [VAI93]; see Figure 10). The weighting of the pixels results from the multiplication of the pixel values and specific *window weights* at the associated window locations.

Mathematically, this procedure is described as follows: Let B be an image which is $N_x \times N_y$ pixels in size with pixel values $B(x, y)$ (intensity, gray values) at horizontal position $x \in \{0,\ldots,N_x-1\}$ and vertical position $y \in \{0,\ldots,N_y-1\}$. A $r \times r$ *window* $F_{i,j}$ at position (i, j), $i \in \{0,\ldots,N_x-1\}$, $j \in \{0,\ldots,N_y-1\}$, is defined by its *window weights* $F_{i,j}(x, y)$:

$$F_{i,j}(x, y) = \begin{cases} 1 & \text{iff } 0 \le x - i < r \text{ and } 0 \le y - j < r \\ 0 & \text{else} \end{cases} \qquad (8)$$

[3] In contrary to the PCA transform $\mathbf{K}_m^T$ the rows $\mathbf{b}_i$ of the ICA transform $\mathbf{B}$ are in general not normalized. The *virtual variance* $var^*(y_i) \equiv \| \mathbf{b}_i \|^{-2}$ of an ICA component $y_i = \mathbf{b}_i \cdot \mathbf{x}$ equals the variance of y_i if $\mathbf{b}_i$ will be normalized.

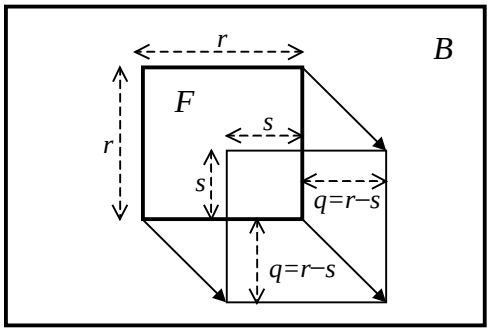


Figure 10: Schematic visualization of the sliding window technique. A window F ($r \times r$ pixel in size) is moved over the image B by $q = r - s$ pixels. The content of F defines a subimage located at the corresponding window position. Note that the subimages overlap by s pixels; if $s = 0$ the subimages are non-overlapping.

The corresponding subimage $U_{i,j}$ at position (i, j) results from the application of $F_{i,j}$ to B:

$$U_{i,j}(x, y) = F_{i,j}(x, y) \cdot B(x, y) \tag{9}$$

Obviously, $U_{i,j}(x, y)$ is non-zero at most at those positions where the window weights $F_{i,j}(x, y)$ are equal to unity; the associated $r \times r$ pixels define the "content" of the window and constitute the coefficients of an $r \cdot r$-dimensional image vector $\mathbf{x}$. The remaining pixels values of $U_{i,j}(x, y)$ are equal to zero and will be ignored.

Equation (8) defines a rectangular window (see Figure 11a): The pixels inside the window are weighted with unity. Another window type which is often used in image processing tasks is the *Gaussian window* (see Figure 11b):

$$G_{i,j,\sigma}(x, y) = F_{i,j}(x, y) \cdot exp\left(-\frac{1}{2\sigma^2} \cdot \left[\left(x - i - \frac{r-1}{2}\right)^2 + \left(y - j - \frac{r-1}{2}\right)^2\right]\right) \tag{10}$$

In contrary to the rectangular window, the pixels inside the Gaussian window are weighted with a two-dimensional Gaussian function. Here, the parameter σ defines the "width" of the Gaussian.

The Gaussian weighting of the subimages has several advantages: Center pixels are emphasized and discontinuities at the borders of subimages causing tiling effects (as known from JPEG or PCA encoded images; see Figure 16b) can be avoided.

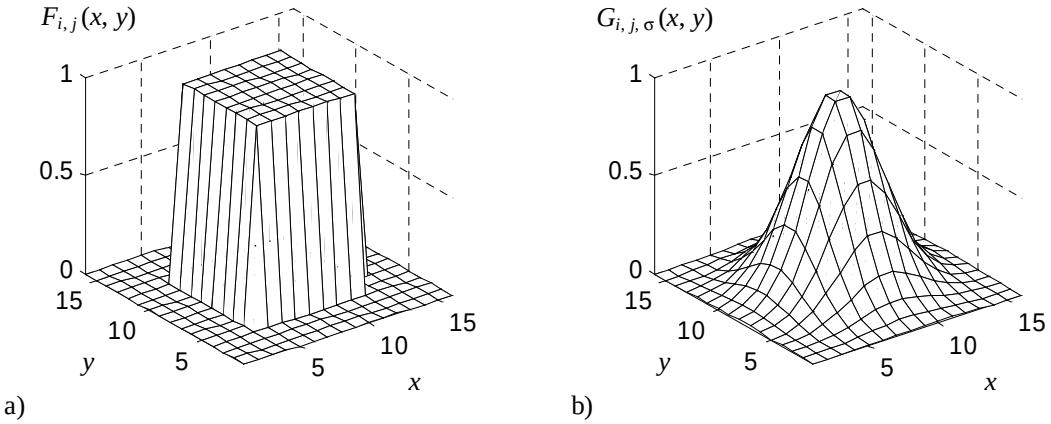


Figure 11: a) Rectangular window $F_{i,j}$ and b) Gaussian window $G_{i,j,\sigma}$.

2.3.3 Overlapping Subimages

The use of Gaussian windows requires the subimages to overlap, or else the information of pixels with low window weights near the borders of the subimages is lost by superimposed background noise. For instance, during the generation of the subimages all pixel values $B(x, y)$ of the image B should be weighted at least with a minimum window level c. The *c-support* of a Gaussian window $G_{i,j,\sigma}$ defined by

$$support(G_{i,j,\sigma}, c) = \{ (x, y) \mid G_{i,j,\sigma}(x, y) \geq c \} \quad (11)$$

consists of the positions (x, y) with associated window weights $G_{i,j,\sigma}(x, y)$ which are at least as large as c. Thus, the c-support represents a circular conic section of $G_{i,j,\sigma}$.

If the Gaussian windows (and with it the generated subimages) do not overlap, the requirement of a minimum weighting stated above will not be met (see Figure 12a). Thus, the Gaussian windows have to overlap (see Figure 12b).

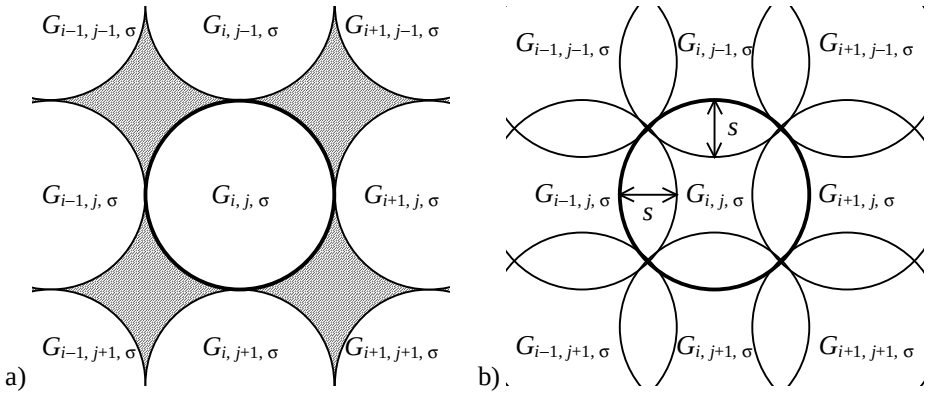


Figure 12: The c-supports of several Gaussian windows. If the windows do not overlap (a), some of the pixels of the image B will be weighted with a value smaller than c (shaded area). If the windows do overlap (b), this case will be excluded.

Arranging the Gaussian windows in a lattice structure according to Figure 12, the windows have to overlap in horizontal and vertical direction by s pixels respectively. Given the width r of the windows, the variance parameter σ, and the minimum weight factor c, it can be shown that the overlap parameter s is

$$s = \left\lceil r - 2 \cdot \sigma \cdot \sqrt{-ln\,(c)} \right\rceil \quad (12)$$

2.3.4 Image Reconstruction

Regardless if the subimages do overlap or not, their reconstruction follows the same principle: First the centered image vectors $\mathbf{x'}$ are reconstructed by the (pseudo) inverse transform $\mathbf{A}^{\dagger}$ and the encoded vectors ξ according to section 2.2.1. In the next steps, the average $\langle \mathbf{x} \rangle$ is added and the image vectors are converted into subimages.

In the case of non-overlapping subimages weighted by unity, the whole image B is derived by merging the subimages at their original location. Mathematically, this is defined in analogy to equation (9) by the addition of subimages $U_{i,j}$:

$$B(x, y) = \sum_{i,j} F_{i,j}(x, y) \cdot B(x, y) = \sum_{i,j} U_{i,j}(x, y) \quad \text{where } i \bmod r = 0,\ j \bmod r = 0 \text{ for all } i, j \quad (13)$$

The positions (i, j) of the rectangular windows $F_{i,j}$ and the subimages $U_{i,j}$ are integer multiples of r since the subimages do not overlap.

If the $U_{i,j}$ do overlap and/or are weighted with a Gaussian function, the addition of the $U_{i,j}$ in general will *not* result in the correct image reconstruction. Here, an additional procedure is required. Assuming

that the subimages were generated by a Gaussian window $G_{i,j,\sigma}$ overlapping by s pixels in horizontal and vertical direction respectively, the reconstructed pixel values of the image B are calculated by

$$B(x,y) = \frac{\sum_{i,j} G_{i,j,\sigma}(x,y) \cdot B(x,y)}{\sum_{i,j} G_{i,j,\sigma}(x,y)} = \frac{\sum_{i,j} U_{i,j}(x,y)}{\sum_{i,j} G_{i,j,\sigma}(x,y)} \quad , i \bmod q = 0,\ j \bmod q = 0 \text{ for all } i,j \qquad (14)$$

where $q = r–s$. Thus, the sum of the subimages $U_{i,j}$ has to be divided by the sum of the windows. Note that equation (14) holds for both overlapping Gaussian and overlapping rectangular windows.

2.4 Experimental Results

Following the investigations in [ARL97] and [AB98a], the PCA and PICA based on non-overlapping subimages were compared to the GPCA and GPICA (*Gaussian-weighted GPCA/GPICA*) based on overlapping, Gaussian-weighted subimages.

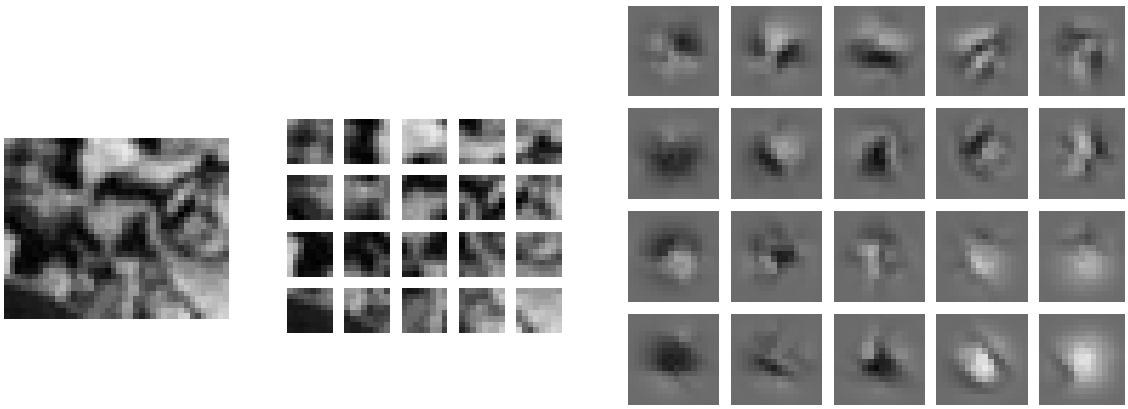
2.4.1 Encoding an Image

The purpose of this comparison was the characterization of the techniques according to their various properties (content of information, compression ratio, reconstruction error). The data consisted of gray scaled images with different content (natural scenes, “artificial” objects such as houses or technical gear, computer-generated graphics); their pixel values (256 gray values) ranged from “0” (black) to “1” (white). For a discussion of the results, the test image *Leaves* (see Figure 13) will be used.

Figure 13: The test image *Leaves.*

According to section 2.3.2, the image was decomposed into two sets of subimages U_R and U_G: U_R consisted of the subimages generated by a rectangular window (8×8 pixels in size). The subimages of U_G were derived from a Gaussian window (16×16 pixels in size) overlapping by 8 pixels in horizontal and vertical direction respectively. The parameter $\sigma = 3$ was empirically chosen to achieve a small reconstruction error (see section 2.4.2). Both U_R and U_G contained about 5100 subimages.

From these sets the image vector sets X_R and X_G were generated according to section 2.2.1. By subtracting the respective average $\langle \mathbf{x}_R \rangle$ and $\langle \mathbf{x}_G \rangle$ the image vectors were centered. Figure 14 shows the corresponding subimages of a small detail of the image *Leaves.*



a) Small detail of *Leaves* (40×32 pixels)

b) Image vectors of X_R (non-overlapping subimages, 8×8 Pixel)

c) Image vectors of X_G (Gaussian-weighted subimages, 16×16 pixels, 8 pixels overlap, $\sigma = 3$)

Figure 14: Decomposition of a small detail (a) into non-overlapping (b) and Gaussian-weighted, overlapping subimages (c).

Next, the covariance matrices $\mathbf{C}_R$ and $\mathbf{C}_G$ of the image vector sets X_R and X_G were calculated. The eigenvectors $\mathbf{e}_{R,i}$ of $\mathbf{C}_R$ represent the PCA primitives of X_R (see section 2.2.2). Consequently, the eigenvectors $\mathbf{e}_{G,i}$ of $\mathbf{C}_G$ are called the *GPCA primitives* of X_G.

After the extraction of the first sixteen principal components, the PICA and GPICA components of X_R and X_G were calculated using the fixed point algorithm in [HO97] (see section 2.3). The rows of the corresponding ICA matrices $\mathbf{B}_R$ and $\mathbf{B}_G$ represent the *PICA* and *GPICA filters* respectively. Since $\mathbf{B}_R$ and $\mathbf{B}_G$ are not invertible, the associated primitives were derived from the columns of the pseudo inverse matrices $\mathbf{B}^\dagger_R$ and $\mathbf{B}^\dagger_G$. According to section 0, the pseudo inverse $\mathbf{B}^\dagger$ of an ICA matrix $\mathbf{B}$ is calculated by $\mathbf{B}^\dagger = \mathbf{W}^\dagger_{\text{pca}} \cdot \mathbf{W}^{\text{T}}_{\text{ica}} = \mathbf{K}_m \cdot \Lambda_{\boldsymbol{m}}^{-½} \cdot \mathbf{W}^{\text{T}}_{\text{ica}}$ where $\Lambda_{\boldsymbol{m}}^{-½} = diag(\lambda_1^{-½},\ldots,\lambda_m^{-½}) = (\Lambda_{\boldsymbol{m}}^{½})^{-1}$.

In Figure 15 the first sixteen PCA/GPCA primitives and PICA/GPICA primitives/filters are shown. Note that it is not possible to calculate the GPCA/GPICA primitives from the PCA/PICA primitives directly. In fact, the statistical properties of subimages generated by a rectangular and a Gaussian window are different; thus the primitives have to be different as well (see also section 2.5).

2.4.2 Quality of Coding and Reconstruction

The image vectors of the sets X_R and X_G were transformed by their corresponding transform matrices. To determine the encoding quality, the expected information content (*marginal entropy*) $H(\xi_i)$ of the resulting components ξ_i of each encoding scheme was calculated [AB98a]:

$$H(\xi_i) = -\sum_{\xi_i} p(\xi_i) \cdot log(p(\xi_i)) \qquad (15)$$

$p(\xi_i)$ represents the occurrence probability of ξ_i. To get comparable results for the different components, the ξ_i were scaled to an identical interval (here: [0, 255]) and rounded to integer values (quantization). This procedure allowed the approximation of the occurrence probabilities by the relative frequencies and the calculation of $H(\xi_i)$ according to equation (15). The sum of the $H(\xi_i)$ of all components (*cumulated entropy*) is a measure for the encoding quality and results in the expected number of *bits* (if the dual logarithm is used in equation (15)) needed for the encoded image vector ξ.

Table 1 shows the cumulated entropy of the different encoding schemes and the expected error (MSE; see equation (2)) of the reconstructed images. The results are similar to those found in [AB98a], i.e. PICA and GPICA have the same reconstruction error but a smaller cumulated entropy than PCA and GPCA. Thus, the compression properties of the PICA and GPICA encoding are slightly better.

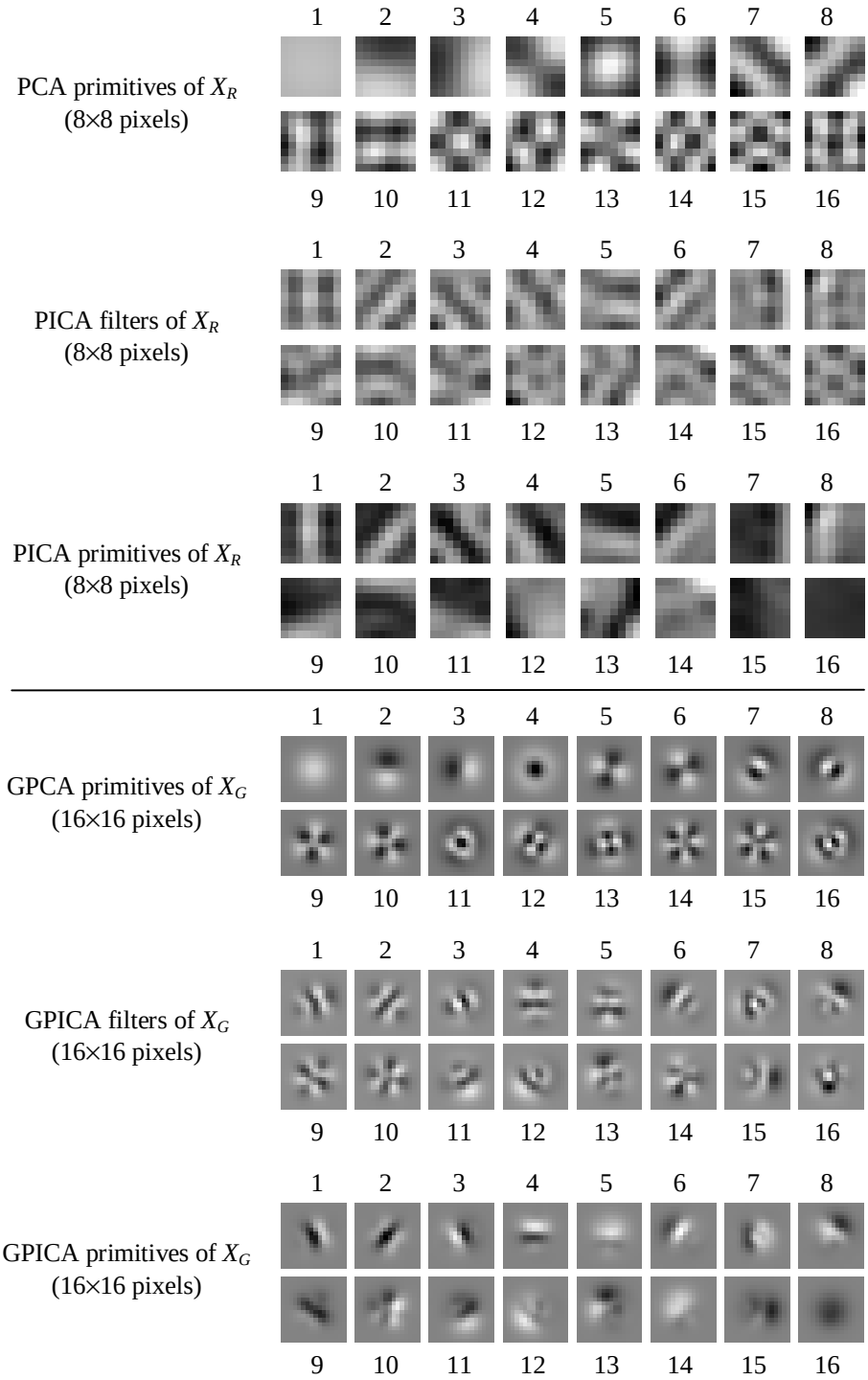


Figure 15: The sixteen PCA/PICA and GPCA/GPICA primitives and filters of the image *Leaves*.

encoding	PCA	PICA	GPCA	GPICA
$\sum_i H(\xi_i)$	104.1 *bits*	100.1 *bits*	104.7 *bits*	100.4 *bits*
MSE	0.0041	0.0041	0.0063	0.0063

Table 1: Cumulated entropy and reconstruction error of the image *Leaves*.

The mean square error of the GPCA/GPICA reconstruction is higher than the one of the PCA/PICA. Furthermore, the GPCA/GPICA reconstructed images appear to be slightly blurred. However, the *subjective* reconstruction error of the GPCA/GPICA perceived by the human visual system is lower: As mentioned in section 2.3.2, tiling effects will be reduced if the image encoding is based on Gaussian-weighted, overlapping subimages.

In Figure 16 the different reconstruction results are presented for the example of a small detail of the image *Leaves* in magnification. To show that the overlapping of the subimages alone does not reduce the tiling effect, an additional PCA encoding was calculated: Here, the subimages were generated in the same way as the subimages of the GPCA/GPICA (16×16 pixels, 8 pixels overlap in horizontal and vertical direction) with the difference of using a *rectangular* window instead of Gaussian window (Figure 16c).

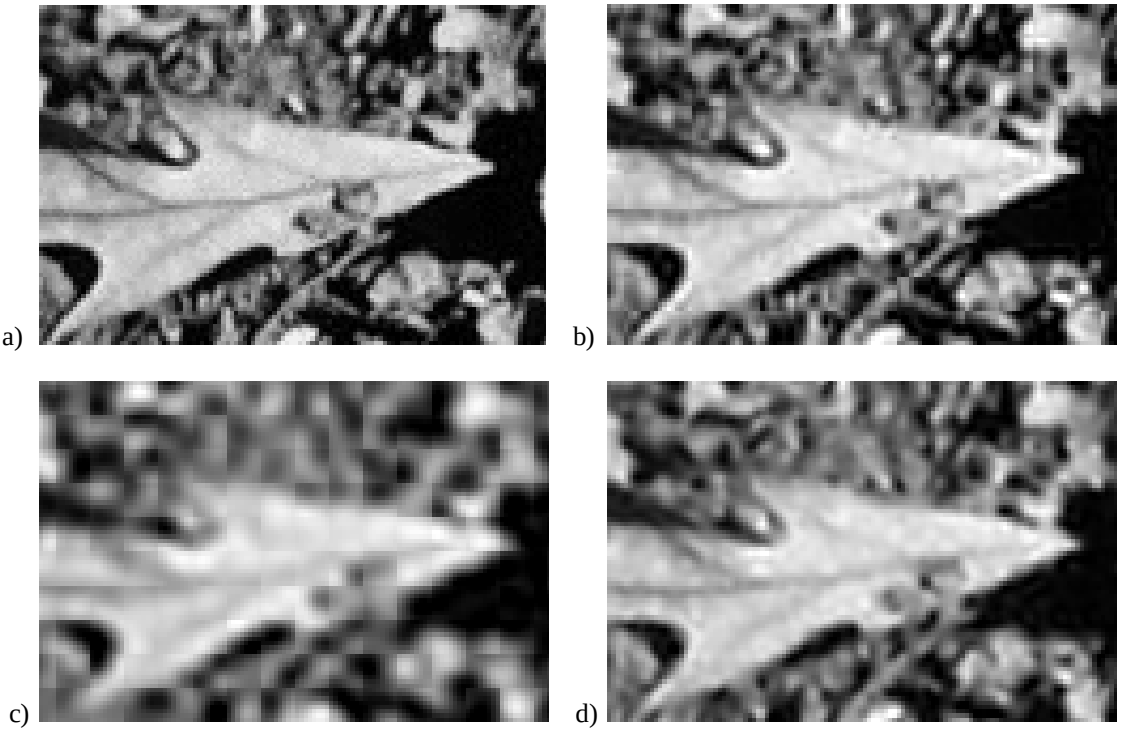


Figure 16: Reconstructing a small detail of the image *Leaves* by sixteen components.
a) Original detail (120×80 pixels),
b) reconstruction by PCA (subimages are 8×8 pixel in size and do not overlap),
c) reconstruction by PCA (subimages are 16×16 pixel in size and overlap by 8 pixels),
d) reconstruction by GPCA (subimages are 16×16 pixel in size and overlap by 8 pixels, $\sigma = 3$).

2.5 Primitives of Scaled Images

One of the biggest problems of automated object recognition tasks is that the size (scale) of objects included in images is unknown. Ideally, objects and images can be encoded independently of their scale. However, the *scale-invariant* encoding of objects and images is impossible until a rather abstract, high-level representation scheme is used (for example the encoding of objects by complex geometric structures). Low-level encoding methods based on the analysis of small subimages are in general not scale-invariant: Images are represented *globally* in the spatial domain by encoded image vectors, even if the image vectors are encoded *locally* in a different domain (e.g. the PCA subspace).

Another question is if the low-level representation of an image and its scaled versions have identical or at least similar properties which are independent of scaling. For example, the amplitudes of the Gabor representation of natural images are proportional to the reciprocal spatial frequency of the corresponding Gabor primitives [FIE87]. This property allows the adaptation of the encoding to the typical frequency distribution of the image data.

Consequently, one might ask if primitives which are adaptively derived for a given image are suitable for scaled versions of the same image as well, or if the corresponding image representations have similar properties. As shown in the following sections, this is true for PCA and GPCA: The primitives remain constant and can be used for a large range of scaled image versions.

The PCA/PICA and GPCA/GPICA primitives of scaled versions of the image *Leaves* were calculated according to section 2.4; Figure 18 shows the resulting GPCA/GPICA primitives. Here, the scaling factor t represents the width or height of the current image version related to the original width or height. Thus, a smaller value of t corresponds to a smaller image size.

In contrary to the PICA/GPICA primitives, the PCA/GPCA primitive sets resemble each other, i.e. they are independent of the size of the image. Further experiments with other image data gave the same results. Some primitives are inverted versions of the primitives of other sets; since the primitives generate a basis of the PCA/GPCA subspace, this is of no importance in case of image encoding tasks.

2.5.1 Analysis of PCA Primitives of Natural Images

The similarity of PCA primitives calculated from different image data was already mentioned in previous work (see for example [FIE87], [SAN89], or [OF96]). However, to us no other work is known which captures the phenomenon of scale-invariant shape of image primitives.

Now, this aspect is investigated in detail. To obtain a suitable image model, Habibi and Wintz give an approximation of the correlation between the gray values $B(x, y)$ and $B(x', y')$ of nearby pixels [HW71]:

$$\rho[B(x, y),\ B(x', y')] \approx exp(-\alpha \cdot |x-x'| - \beta \cdot |y-y'|) = exp(-\alpha \cdot |x-x'|) \cdot exp(-\beta \cdot |y-y'|) \qquad (16)$$

Thus, the approximated correlation function depends only on the parameters $\alpha, \beta > 0$ and the spatial distance of the pixels. Note that the approximated correlation function is *separable*, i.e. the horizontal and vertical terms in equation (16) are independent from each other. Figure 17 shows the actual correlation of the pixels of the image *Leaves* in horizontal direction and the approximated correlation in (16).

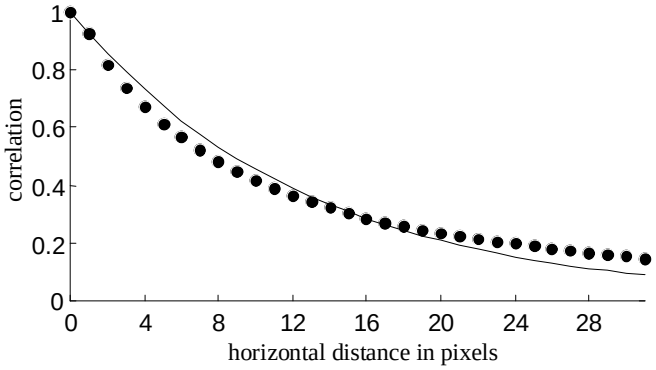


Figure 17: Numerically calculated correlation of pixels of the image *Leaves* (dots) and its approximation (line; $\alpha = 0.0782$) in horizontal direction.

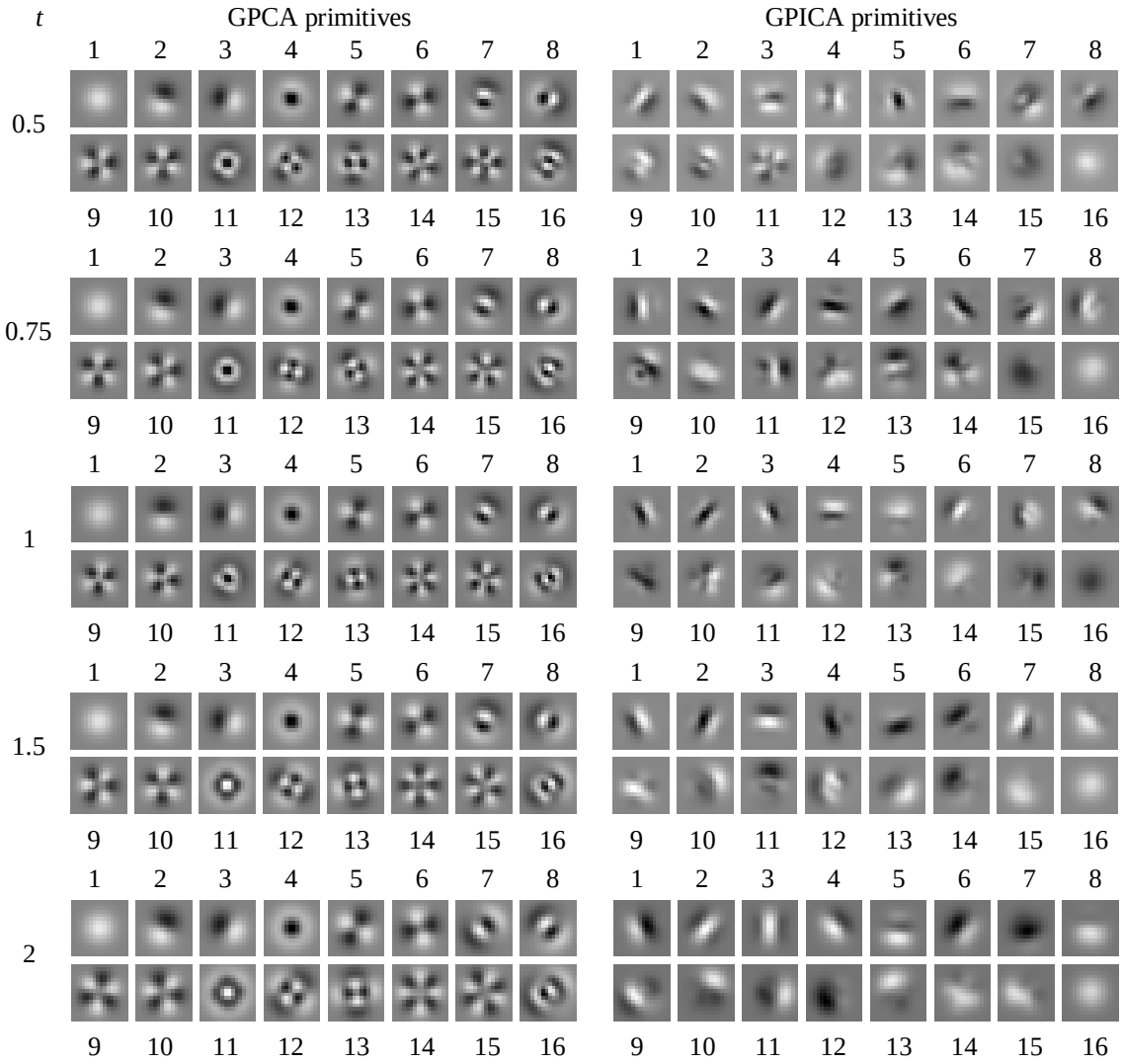


Figure 18: GPCA and GPICA primitive sets of scaled versions of the image *Leaves*. For a large scaling range (t = 0.5 … 2), the GPCA primitive sets are almost identical. In some sets, primitives no.7 to no.16 are inverted; this is of no importance for image encoding tasks. In contrary, the GPICA primitive sets are not identical.

From equation (16) and the standard deviations $\sigma[B(x, y)]$ and $\sigma[B(x', y')]$ of the pixels, the covariance can be approximated:

$$cov[B(x, y),\ B(x', y')] \approx \sigma[B(x, y)] \cdot \sigma[B(x', y')] \cdot exp(-\alpha \cdot |x-x'|) \cdot exp(-\beta \cdot |y-y'|) \qquad (17)$$

In the following, natural images are assumed to have a covariance given by equation (17). Furthermore, their associated PCA primitives are calculated from subimages which are $r \times r$ pixels in size and generated by a rectangular window. Because of the uniform window weights, the standard deviations of the pixels are equal, i.e. $\sigma[B(x, y)] = \sigma[B(x', y')] = \sigma_B$. Thus, the covariance function in equation (17) differs from the correlation function in equation (16) only by the constant factor σ_B.

In this case the PCA primitives can be represented as two-dimensional *eigenfunctions* $\varphi_{\alpha,\beta}(x, y)$ of the covariance function with *eigenvalues* $\lambda_{\alpha,\beta}$ and $x, y \in \{0,\ldots, r-1\}$ [4]. $\varphi_{\alpha,\beta}(x, y)$ satisfies the condition

[4] Here, the covariance function $cov[B(x, y), B(x', y')]$ and the eigenfunctions $\varphi_{\alpha,\beta}(x, y)$ correspond to the covariance matrix $\mathbf{C}$ and the eigenvectors $\mathbf{e}_i$ respectively (see section 2.2.2).

$$\sum_{x'=0}^{r-1}\sum_{y'=0}^{r-1} cov[B(x, y),\ B(x', y')] \cdot \varphi_{\alpha,\beta}(x', y') \ = \ \lambda_{\alpha,\beta} \cdot \varphi_{\alpha,\beta}(x, y) \qquad (18)$$

Because of the separability of the covariance function $cov[B(x, y),\ B(x', y')]$, $\varphi_{\alpha,\beta}(x, y)$ is given by the product of two one-dimensional eigenfunctions $\varphi_\alpha(x)$ and $\varphi_\beta(y)$; the eigenvalues $\lambda_{\alpha,\beta}$ are the product of the eigenvalues λ_α and λ_β of $\varphi_\alpha(x)$ and $\varphi_\beta(y)$. This results from equations (17) and (18) combined with the assumption $\sigma[B(x, y)] = \sigma[B(x', y')] = \sigma_B$:

$$\lambda_{\alpha,\beta} \cdot \varphi_{\alpha,\beta}(x, y) \ = \ \lambda_\alpha \cdot \lambda_\beta \cdot \varphi_\alpha(x) \cdot \varphi_\beta(y)$$

$$\text{where} \qquad \sum_{z'=0}^{r-1} \sigma_B \cdot exp(-\gamma \cdot |\, z - z'\,|) \cdot \varphi_\gamma(z') \ = \ \lambda_\gamma \cdot \varphi_\gamma(z) \qquad (19)$$

Here, the parameter γ and the variable z are placeholders for α, β and x, y respectively. The solution of equation (19) is given by

$$\varphi_\gamma(z) \ = \ a \cdot cos\left(b \cdot \gamma \cdot \left(z - \tfrac{r-1}{2}\right)\right) \qquad \text{and} \qquad \varphi_\gamma(z) \ = \ a' \cdot sin\left(b \cdot \gamma \cdot \left(z - \tfrac{r-1}{2}\right)\right) \qquad (20)$$

(see [HW71] [DR87] [BRA95]). a and a' are suitable constants to normalize $\varphi_\gamma(z)$; the parameters b and b' have to be calculated numerically to satisfy the equations

$$b \cdot tan(\tfrac{1}{2} \cdot b \cdot \gamma \cdot r) = 1 \qquad \text{and} \qquad b' \cdot cot(\tfrac{1}{2} \cdot b' \cdot \gamma \cdot r) = -1 \qquad (21)$$

The eigenvalues are

$$\lambda_\gamma \ = \ \frac{2}{\gamma \cdot (b^2 + 1)} \qquad \text{and} \qquad \lambda_\gamma \ = \ \frac{2}{\gamma \cdot (b'^2 + 1)} \qquad (22)$$

The shape of an eigenfunction $\varphi_\gamma(z)$ in equation (20) is given by a trigonometric function with frequency f_γ; here, the frequency is equal to the products $f_\gamma = b \cdot \gamma$ or $f_\gamma = b' \cdot \gamma$. The eigenfunctions with the smallest possible parameters b or b' have the largest eigenvalues λ_γ. Thus, the first, "most salient" PCA primitives have the smallest possible spatial frequency, since the principal components are ordered according to their descending eigenvalues.

To determine the influence of the size of a given image B on the frequency f_γ of the eigenfunctions $\varphi_\gamma(z)$ (i.e. the shape of the PCA primitives), the discrete lattice structure of the pixels is ignored. In this case, the pixel values of an image B_t, which is derived by scaling the image B with a factor t, are calculated by

$$B_t(x, y) \ = \ B\left(\frac{x}{t}, \frac{y}{t}\right) \qquad (23)$$

Applied to equation (16), the correlation of the pixels is given by

$$\rho[B_t(x, y),\ B_t(x', y')] \ \approx \ exp(-t^{-1} \cdot \alpha \cdot |\, x - x'\,|) \cdot exp(-t^{-1} \cdot \beta \cdot |\, y - y'\,|) \qquad (24)$$

Note that the reciprocal of the scaling factor t is multiplied with the original parameters α and β. Thus, the new model parameters of the scaled image B_t are equal to the products $-t^{-1} \cdot \alpha$ and $-t^{-1} \cdot \beta$, and the frequency $f_{t \cdot \gamma}$ of the corresponding eigenfunctions $\varphi_{t \cdot \gamma}(z)$ are calculated by

$$f_{t \cdot \gamma} = b \cdot t^{-1} \cdot \gamma \ , \ b \cdot tan(\tfrac{1}{2} \cdot b \cdot t^{-1} \cdot \gamma \cdot r) = 1 \qquad \text{and} \qquad f_{t \cdot \gamma} = b' \cdot t^{-1} \cdot \gamma \ , \ b' \cdot cot(\tfrac{1}{2} \cdot b' \cdot t^{-1} \cdot \gamma \cdot r) = -1 \qquad (25)$$

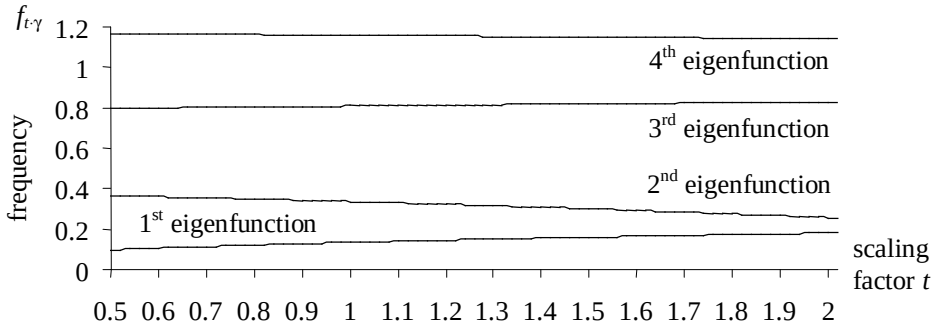


Figure 19: The frequency $f_{t\cdot\gamma}$ of the first four eigenfunctions $\varphi_\gamma(z)$ in equation (20) with largest eigenvalues λ_γ as a function of the scaling factor t (r = 8, γ = 0.0782).

If the size of the subimages is fixed, the frequency of the eigenfunctions will remain almost constant for a large scaling range (see Figure 19): The frequency shift caused by the scaling factor t will be compensated by the parameters b and b'. This property explains the similarity of the PCA primitives derived from versions of the same image which differ in size (compare to Figure 20).

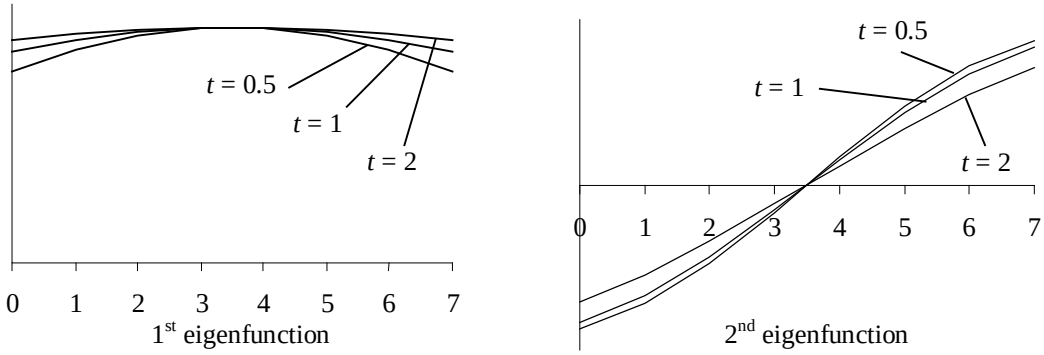


Figure 20: The first two eigenfunctions $\varphi_\gamma(z)$ in equation (20) with the largest eigenvalues as a function of the scaling factor t (r = 8, γ = 0.0782).

2.5.2 Analysis of GPCA Primitives of Natural Images

In the previous section, the PCA primitives were represented by the product of two one-dimensional eigenfunctions $\varphi_\alpha(x)$ and $\varphi_\beta(y)$. For GPCA primitives, a corresponding representation is given by the product of the eigenfunctions $\varphi'_\alpha(x)$ and $\varphi'_\beta(y)$. However, these eigenfunctions do not posses an analytical formulation comparable to equation (20) and have to be numerically approximated. The reason is, that the product of the standard deviations of the pixel values in equation (17) is no longer a constant factor σ_B, but (because of the Gaussian weighting function) equals to

$$\sigma[B(x,y)] \cdot \sigma[B(x',y')] = \sigma_B \cdot g_{r,\sigma}(x,y) \cdot g_{r,\sigma}(x',y')$$

$$\text{where} \qquad g_{r,\sigma}(x,y) = exp\left[-\tfrac{1}{2}\cdot\sigma^{-2}\cdot\left(\left(x-\tfrac{r-1}{2}\right)^2+\left(y-\tfrac{r-1}{2}\right)^2\right)\right] \qquad (26)$$

Note that the parameter σ represents the width of the Gaussian (see section 2.3.2). Combining equations (17), (18) and (26), the eigenfunctions $\varphi'_\alpha(x)$, $\varphi'_\beta(y)$ and their corresponding eigenvalues λ'_α, λ'_β have to satisfy the condition (compare to equation (19) for the PCA)

$$\sum_{z'=0}^{r-1} \sigma_B \cdot exp\left[-\tfrac{1}{2}\cdot\sigma^{-2}\cdot\left(\left(z-\tfrac{r-1}{2}\right)^2+\left(z'-\tfrac{r-1}{2}\right)^2\right)\right]\cdot exp(-\gamma\cdot|z-z'|)\cdot\varphi'_\gamma(z') \;=\; \lambda'_\gamma\cdot\varphi'_\gamma(z) \qquad (27)$$

Again, γ and z are placeholders for α, β and x, y respectively. According to the latest findings, an analytical formulation of the eigenfunctions $\varphi'_\gamma(z)$ similar to equation (20) can not be given. However, as shown by a numerical analysis, the $\varphi'_\gamma(z)$ will vary even less than the eigenfunctions $\varphi_\gamma(z)$, if the scaling parameter t is changed (see Figure 21). Furthermore, the eigenfunctions $\varphi'_\gamma(z)$ with largest eigenvalues resemble the derivatives of a Gaussian function which are used as filters in Scale-Space theory [LIN94].

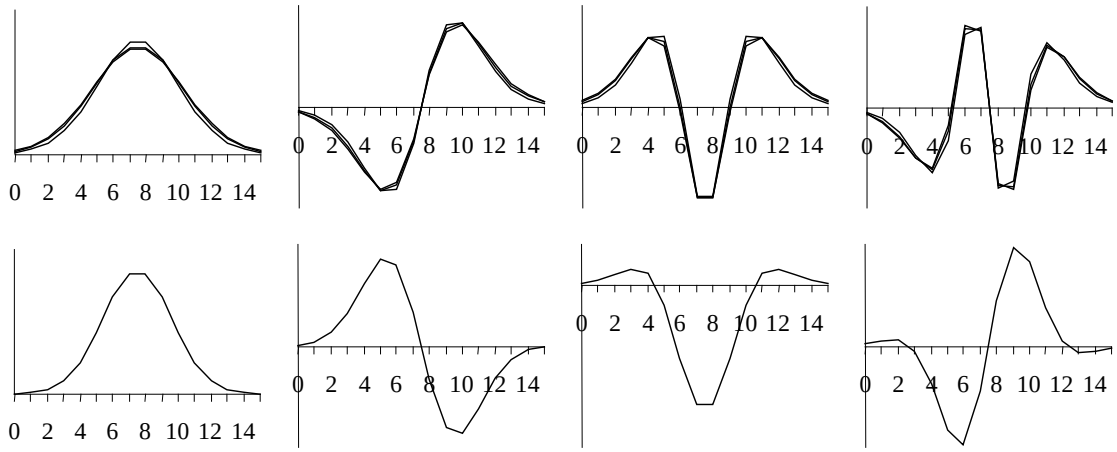


Figure 21: (above) The first four eigenfunctions $\varphi'_\gamma(z)$ for t = 0.5, 1, and 2 respectively.
(below) A Gaussian function and its first three derivatives.

2.6 Generalized GPCA Primitives

The analytical and experimental results of the previous section suggest the possibility of using only one set of image primitives to encode scaled versions of an image as well as images which are different from each other. Obviously, images with similar correlation parameters α and β in equation (16) have similar eigenfunctions $\varphi_\gamma(z)$ and $\varphi'_\gamma(z)$.

The advantage of *generalized* GPCA primitives (i.e. primitives suitable for different images) is, that only the parameters α, β and the numerical solutions of equation (27) have to be calculated to generate the primitives. Furthermore, GPCA primitives determined for certain parameters α and β can be reused to encode images with similar correlation parameters.

At the beginning of section 2.5.1, some references were given concerning the similarity of PCA primitives of different image data; the results therein could be verified for GPCA primitives as well.

Figure 22 shows the GPCA primitives of three different images and the two-dimensional eigenfunctions $\varphi'_{\alpha,\beta}$ calculated for the image *Leaves*. At first sight, the primitive sets seem to be completely different; however, most of the primitives appear as rotated versions in varying order.

Particularly, the eigenfunctions $\varphi'_{\alpha,\beta}$ are more similar to the GPCA primitives of the images *Africa* and *Room* than to the primitives of the image *Leaves* themselves. The reason is that in equation (16) the largest correlation is assumed along the horizontal and vertical axes of the image (see Figure 23a). This assumption holds for *Africa* and *Room*, whereas the axes of largest correlation of *Leaves* are slightly rotated against the horizontal and vertical image axes (see Figure 23b). Consequently, the GPCA primitives are rotated as well.

A detailed analysis of these effects will be subject of future investigations. Possibly, a more precise approximation of the correlation function could be derived by modeling the rotation of the largest correlation axes with the aid of a parameter ϕ which represents the rotation angle.

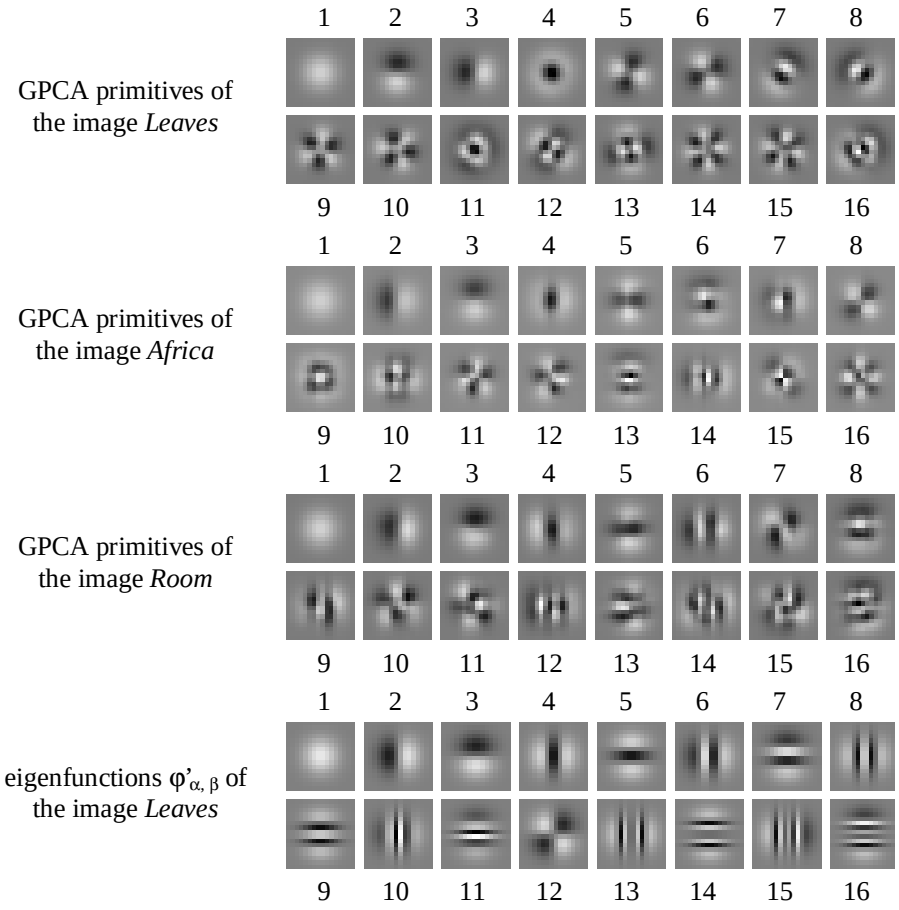


Figure 22: GPCA primitives (16×16 pixels in size) of three different images and the eigenfunctions $\varphi'_{\alpha,\beta}$ of the image *Leaves*.

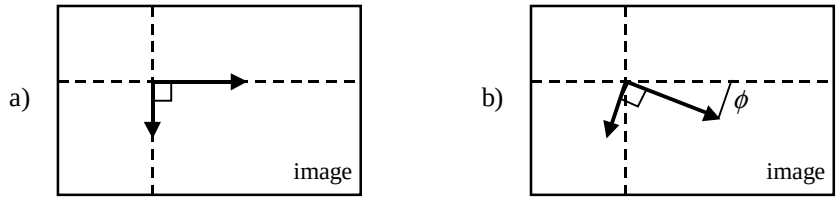


Figure 23: a) The axes of the largest pixel correlation in horizontal and vertical direction. (b) The axes of pixel correlation rotated by the angle ϕ as compared to (a).

2.7 Discussion

The investigations so far have resulted in three novel methods for encoding images based on the transform coding principle:

1) PICA encoding with minimum information content and minimum reconstruction error,
2) GPCA encoding by Gaussian-weighted subimages and minimum tiling effects, and
3) GPICA encoding which combines PICA and GPCA.

The advantages of these methods compared to other encoding strategies are:

- adaptive image primitives, i.e. primitives which are adjusted to the characteristics of the respective image data;
- decorrelated (in case of the GPICA even statistical independent) components allowing an efficient encoding and compression of images;
- low reconstruction error and a minimum of generated artifacts (tiling).

Further advantages of the GPCA compared to the GPICA are:

- fast generation of the primitives since the ICA stage is omitted;
- identical or similar primitives for scaled versions of an images or for different images with similar statistical properties;
- possibility of efficient calculation based on a reliable image model.

Because of the properties mentioned last, the analytically modeled GPCA is superior to the numerically determined GPICA.

What are the drawbacks of the developed methods? A typical drawback of adaptive image encoding strategies compared to static methods like DFT, DCT, or DGT is that in theory the image primitives have to be recalculated for each image. However, the GPCA primitive sets of scaled images or different images with similar correlation parameters resemble each other, and can therefore be reused as well.

Thus, if images are classified according to their correlation parameters α and β, the images of one class and their scaled versions will be encoded by a common set of primitives which is calculated only once. The reconstruction error of images with different parameters is slightly higher, but negligible for practical purposes. Furthermore, it can be compensated by simple methods (e.g. reducing the degree of compression).

3 Object Recognition

Starting with a simple standard technique, the *template matching*, the recent steps in developing an object recognition system for digital images are described in this chapter. The investigations concentrate on the required invariance against scaling of the different methods; this invariance is a necessary condition to recognize objects successfully.

3.1 Scale-invariant Object Recognition

In chapter 1 the most important problems of the automated object recognition were presented. The research within the scope of the project SEMACODE focuses on the problem of a scale-invariant recognition which is not solved in a satisfactory manner by already existing concepts.

Why is an efficient scale-invariant recognition of objects difficult? One of the main reasons is the discrete structure of the image data: Images showing the same object but differing in size have a different number of pixels. The number of pixels determines the resolution of images; the smaller the image, the lower the resolution. The resolution of an image determines the quality of the representation of an object and its details. Consequently, the classification of images showing the same object but differing in size becomes ambiguous. Furthermore, the scaling of images showing different objects may result in identical low-detailed images. This is demonstrated in Figure 24.

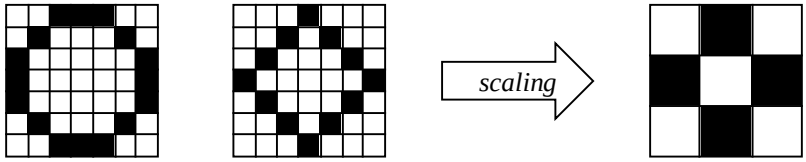


Figure 24: Example for the generation of an identical image by scaling two different images.

On the other hand, too many details may lead to erroneous recognition results. Humans have an intuitive notion which details are “essential” to characterize an object and which details can be neglected. Additionally, humans are able to decide reliably if a detail belongs to an object or if it is a “disturbance” (i.e. the detail of different object, a defect of the image, noise, etc.). Similar mechanisms to recognize objects automatically do only exist in rudimentary form.

Thus, to obtain reasonable results, the recognition task has to be limited to a certain scaling range which depends on the given object. The problem is to determine this scaling range, and how to use it efficiently in the search process.

3.1.1 Existing and New Methods

Starting point of the investigations and developments was a classical, not scale-invariant technique, the *template matching* [PRA78] [GW93]. It is based on the main idea of determining those parts of an image most *similar* to the given object image. Since the exact recognition of an object is often impossible (see chapter 1), the search for similar objects is the only reasonable alternative.

Many concepts exist to define a quantitative measure for the similarity of images; well-known are the Euclidean distance and the correlation coefficient. Other measures can be found for example in [SJ99]. The current research concentrates mostly on the use of the correlation because of its advantageous properties and its intuitive correspondence to the subjective concept of “similarity”. Future investigations will include other similarity measures as well.

As an expansion of the classical template matching, the *blurred template matching* was developed by us, which suites a limited scale-invariant object recognition. A variant of this approach integrates the GPCA encoding of images which was introduced in chapter 2.

The main idea is to reduce the image data to a few salient details, the so-called *points of interest* (PoI) of images. Their usage in an attention-based object recognition system were investigated and are

described in this section. Furthermore, these points are the basis of an object or image model which is ideally scale-invariant. First results show that this model can be used to successfully recognize objects.

3.2 Template Matching

Our research started with the investigation of a classical technique, the template matching, a very simple and reliable method to recognize objects in images. However, it is assumed that the size of the object is known, and that the object is at most slightly distorted or occluded by other objects. Especially the missing invariance against scaling restricts the usage of the standard template matching. Although it is possible to search for a version of the given object only differing in size, the resulting computational complexity is not acceptable for most recognition tasks.

Thus, an expanded version of the template matching, the blurred template matching, was developed. Here, both the object and the image are reduced in size to decrease the computational complexity and to allow an "imprecise" comparison between the object and parts of the image: Because of the lower image resolution, unimportant details are omitted, and the recognition process becomes more robust against small scaling, deformation, or occlusion effects. However, this might lead to a recognition of objects which are *not* included in the current image.

In further investigations the template matching was combined with the GPCA encoding described in chapter 2. Compared to the blurred template matching, similar results were achieved while the degree of "blurring" could be regulated more precisely.

3.2.1 Classical Template Matching

In the following, an object is assumed to be given by an object image *Obj* which is $n_x \times n_y$ pixels in size. The classical *template matching* (see for example [PRA78] or [GW93]) is based on the idea of moving a $n_x \times n_y$ window F over the *target image* B which has to be processed (c.f. the sliding window technique described in section 2.3.2). Mathematically, the pixels inside the window F at position (i, j) generate a subimage $U_{i,j}$ which is $n_x \times n_y$ pixels in size. The pixel values $U_{i,j}(x, y)$ of the subimage are calculated according to equation (9) by the product of the weights $F_{i,j}(x, y)$ of the rectangular window and the pixel values $B(x, y)$ of B, i.e. $U_{i,j}(x, y) = F_{i,j}(x, y) \cdot B(x, y)$.

During every processing step of the template matching, the correlation coefficient $\rho_{i,j}(U_{i,j}, Obj)$ of the current subimage $U_{i,j}$ at position (i, j) and the pixel values $Obj(x, y)$ of the object image (the *template*) is calculated:

$$\rho_{i,j}(U_{i,j}, Obj) = \frac{cov(U_{i,j}, Obj)}{\sqrt{var(U_{i,j})} \cdot \sqrt{var(Obj)}}$$

$$\text{where } cov(U_{i,j}, Obj) = \frac{1}{n_x \cdot n_y - 1} \cdot \sum_{x=0}^{n_x-1} \sum_{y=0}^{n_y-1} U_{i,j}(i+x, j+y) \cdot Obj(x,y) - \overline{U_{i,j}}(i+x, j+y) \cdot \overline{Obj}(x,y),$$

$$\overline{U_{i,j}}(x,y) = \frac{1}{n_x \cdot n_y} \cdot \sum_{x=0}^{n_x-1} \sum_{y=0}^{n_y-1} U_{i,j}(i+x, j+y), \qquad \overline{Obj}(x,y) = \frac{1}{n_x \cdot n_y} \cdot \sum_{x=0}^{n_x-1} \sum_{y=0}^{n_y-1} Obj(x,y)$$

$$\text{and} \quad var(U_{i,j}) = \frac{1}{n_x \cdot n_y - 1} \cdot \sum_{x=0}^{n_x-1} \sum_{y=0}^{n_y-1} \left[U_{i,j}(i+x, j+y) - \overline{U_{i,j}}(i+x, j+y) \right]^2,$$

$$var(Obj) = \frac{1}{n_x \cdot n_y - 1} \cdot \sum_{x=0}^{n_x-1} \sum_{y=0}^{n_y-1} \left[Obj(x,y) - \overline{Obj}(x,y) \right]^2 \qquad (28)$$

Possible values of the correlation coefficient $\rho_{i,j}(U_{i,j},Obj)$ are derived from the interval [−1,1]. According to the current value of $\rho_{i,j}(U_{i,j},Obj)$, one of the following statements related to the similarity of the subimage and the object image holds:

$\rho_{i,j}(U_{i,j},Obj) = 1$:	object and subimage are identical up to a constant factor $c_1>0$ and an additive constant c_2, i.e. $U_{i,j}(i+x, j+y) = c_1 \cdot Obj(x, y) + c_2$
$0 < \rho_{i,j}(U_{i,j},Obj) < 1$:	the higher the value of the correlation coefficient, the more similar are $U_{i,j}$ and *Obj*
$\rho_{i,j}(U_{i,j},Obj) = 0$:	object and subimage do not resemble each other
$-1 < \rho_{i,j}(U_{i,j},Obj) < 0$:	the lower the correlation coefficient, the more similar are $U_{i,j}$ and the inverted image of *Obj*
$\rho_{i,j}(U_{i,j},Obj) = -1$:	subimage and inverted object image are identical up to a constant factor $c_1>0$ and an additive constant c_2, i.e. $U_{i,j}(i+x, j+y) = -c_1 \cdot Obj(x, y) + c_2$

The advantages of the correlation coefficient are its distinctiveness and its invariance against the intensity and average illumination of the object image and the subimages. Furthermore, the inverse object image can be recognized as well.

A drawback is the high computational complexity, especially if the size of the object appearing in the image is unknown: in this case the correlation coefficient has to be calculated for many versions of the object image which differ in their size.

3.2.2 Decreasing the Resolution of Images

To decrease the computational complexity of the template matching, both the object image and the target image were reduced in size. Furthermore, the robustness of the processing task against deformation, distortion, scaling, and occlusion is expected to be increased, since the resulting lower resolution and the corresponding "blurring" of the images should lead to a faster and less restrictive object recognition.

The correctness of these assumptions was verified using a few simple objects; the results are demonstrated for the object images *Stool*, *Stool-O*, and *Container* (see Figure 25).

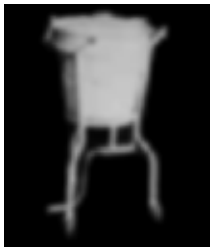
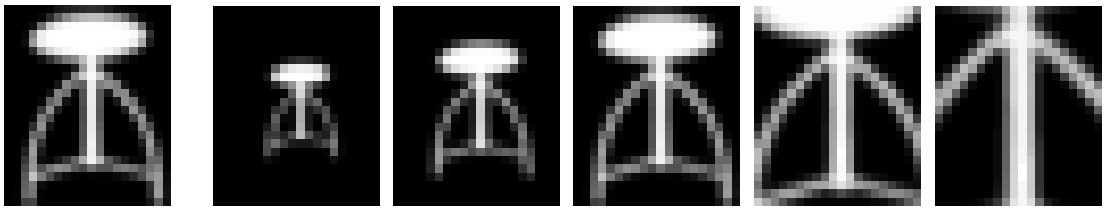
a)  b)  c) 

Figure 25: The object images *Stool* (a), *Stool-O* (b) and *Container* (c).

First, several versions of the image *Stool* (107×126 pixels) differing in their size were calculated (scaling factor $t = 0.2 \ldots 5$). Then the image and its scaled versions were reduced to the same size (for example 20×24 pixels; this corresponds to a scaling factor[5] of 0.19 in case of the original image). Figure 26 shows some of the resulting images. At different image resolutions, the correlation coefficient of the original image and the scaled versions as a function of the scaling factor t is reproduced in Figure 27.

[5] This factor is called the *resolution factor* t_R, see next section (3.2.3).



a) original (20×24 pixels) | *t* = 0.5 | *t* = 0.7 | *t* = 1 | *t* = 1.4 | *t* = 2

b) versions of (a) scaled by the factor *t* and reduced to 20×24 pixels

Figure 26:The reduced image *Stool* (a) and its reduced, scaled versions (b).

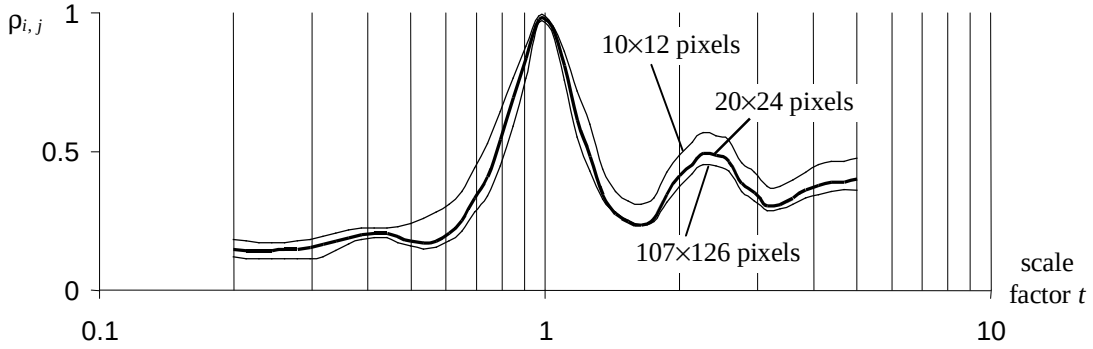


Figure 27: The correlation coefficient of the image *Stool* and its scaled versions as a function of the scaling factor *t* at different resolution levels.

As expected, the correlation coefficient reaches its maximum value of unity at $t = 1$, since in this case the compared images are equal. The correlation coefficients at a resolution of 20×24 pixels and the original resolution of 107×126 pixels are nearly identical; in fact, a significant difference can not be found unless the resolution is reduced to 10×12 pixels. Thus, lowering the image resolution up to a certain degree does not affect the results, while the computational complexity is decreased: The number of pixel values needed to calculate the correlation coefficient reduces from originally 107×126 = 13482 to 20×24 = 480.

Of particular interest is that the correlation coefficient will remain high if the scaling factor is varied by a small amount: For a limited scaling range, the correlation coefficient is almost scale-invariant.

Next, the correlation coefficient of the image *Stool* and the scaled versions of both a "distorted" image *Stool-O* and a "similar" image *Container* were calculated. In case of *Stool-O*, the "distortion" results from a background object to simulate an occlusion effect. The corresponding correlation coefficients are shown in Figure 28.

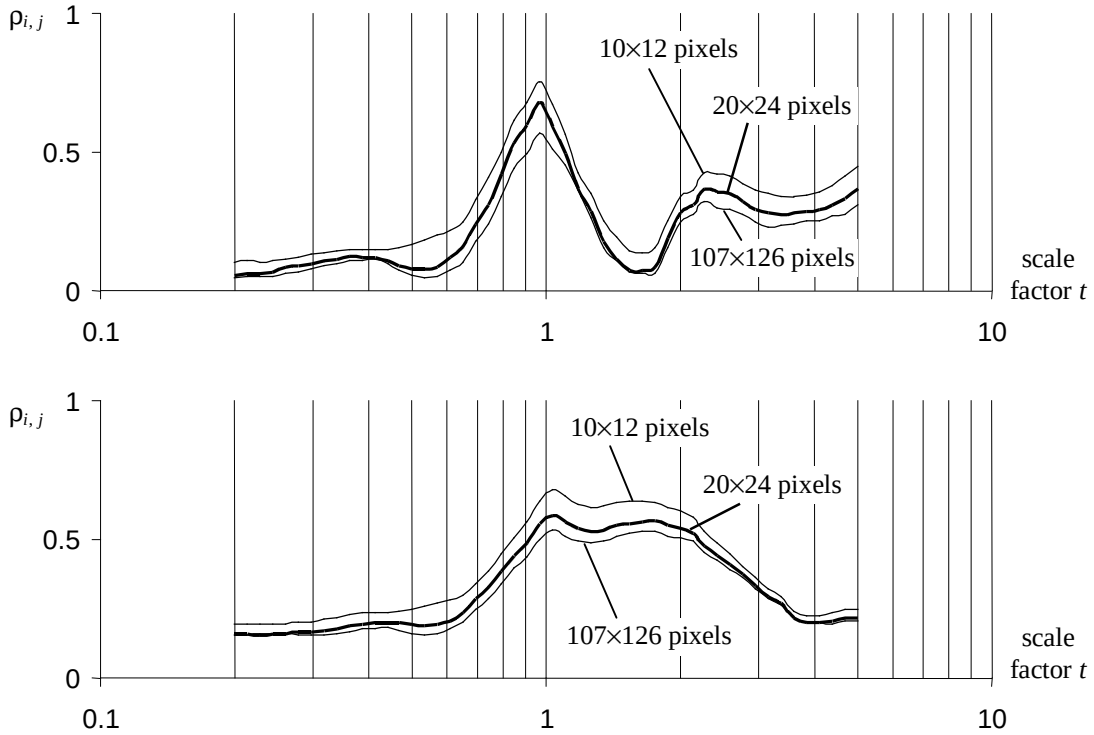


Figure 28: The correlation coefficient of the images *Stool* and the scaled versions of *Stool-O* (above) and *Container* (below) as a function of the scaling factor *t* at different resolution levels.

For the image *Stool-O*, the correlation coefficient reaches its maximum at $t = 0.97$, i.e. the simulated occlusion results both in a lower correlation and a small scaling error. Note that the correlation coefficient is higher for a lower image resolution: This confirms the assumption that the tolerance against deformation and distortion effects will be increased if the image resolution is reduced.

The interpretation of the results derived for the image *Container* is more difficult. At a large scaling range ($t = 1…2$) the value of the correlation coefficient is higher than 0.5; the overall maximum is reached at $t \approx 1.1$ indicating that the object shown in the image *Container* is "larger" than the object *Stool*. Thus, both objects are "similar" in terms of the correlation measure. However, the exact scaling factor can not be determined, since the graph of the correlation coefficient in the neighborhood of the maximum value is relatively flat.

Here, the problem is if the two objects in fact are "similar" in terms of human visual perception. This question is difficult to answer since the subjective notion of similarity is based on individual parameters, i.e. a person *A* states that two objects are similar, whereas person *B* does not. The perceived similarity will become less ambiguous if *more* than two objects are compared and a statement like "object No.1 is more similar to object No.2 than object No.3" can be given. Thus, similarity highly depends on the context in which the objects are presented. For a detailed discussion of these aspects see for example [SJ99].

3.2.3 Blurred Template Matching

After the investigations described in the previous section, the template matching with "blurring" (*blurred template matching*), i.e. with reduced image resolution, was used to recognize several objects in a real-world image *B*. Therefore, the size of *B* and the object images was decreased by a factor $t_R << 1$ called the *resolution factor*.

The search for an object in the reduced image B_R was done at different scales $t \in [t_{min}, t_{max}]$, since the correlation coefficient is scale-invariant only for a limited scaling range. Thus, the images of the objects to be recognized were scaled by the product $t \cdot t_R$ of the scaling factor and the resolution factor.

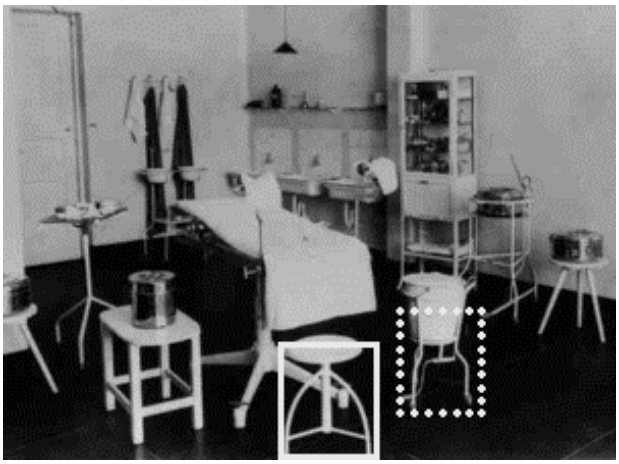


Figure 29: Results of recognizing the object *Stool* in the target image *Room*. The marked image details and the object image have the largest correlation coefficients (solid line: correlation coefficient = 0.75 at t = 0.48; dotted lines: correlation coefficient = 0.72 at t = 0.4).

Figure 29 shows the results of recognizing the object *Stool* in the target image *Room* (301×223 pixels) by using the blurred template matching. The stool visible in *Room* is about the factor 0.5 smaller in size than the object image *Stool*; the size of the image *Room* at reduced resolution was 44×59 pixels (t_R = 0.2). The blurred template matching was performed at ten different scaling ranges (t = 0.4 , 0.48 , 0.57 , 0.68 , 0.82 , 0.98 , 1.17 , 1.4 , 1.67 , and 2.0). Thus, the resolution of the reduced object image ranged from 8×10 to 42×50 pixels.

The image detail with maximum correlation coefficient (= 0.75) was found at a scale of t = 0.48 which is the closest to the exact scaling range 0.5. The location of this image detail matched the location of the stool visible in *Room*. Interestingly, the second largest coefficient (= 0.72) was calculated for the image detail containing the object *Container* (see Figure 29). However, the object *Stool* could not be recognized if none of the scaling ranges t was close to the exact scale of 0.5 .

Besides *Stool*, the objects *Container* and *Container2* (see Figure 30a) were searched in *Room* as well, but only *Container* could be found. Obviously, the "distortion" of the object *Container2* visible in *Room*, which is caused by the background (dark floor and bright wall), prevented a successful recognition.

Further recognition tasks included the object images *Chair* and *Box* (see Figure 30b+c). These images represent rough sketches of the contours of real-world objects and were used to simulate possible object examples generated with the aid of a graphical editor by a hypothetical user. Since the inverted images of these objects might be included in the target image as well, the absolute value of the correlation coefficient was calculated (compare to section 3.2.1). In this case, the chair visible at bottom left in *Room* could be recognized. In contrary, the search for the object *Box* was not successful: In *Room*, every object with a shape similar to the sketched box has textures caused by illumination or reflection, which are not visible in the object image *Box*.

a)  b)  c) 

Figure 30: The object images *Container2* (a), *Chair* (b) and *Box* (c).

To perform the recognition task, the object images as well as the target image have to be reduced in size to achieve a lower image resolution. Does the utilization of scale-invariant GPCA primitives improve this process? This will be investigated in the next section.

3.2.4 Template Matching of GPCA-encoded Images

So far, to recognize objects by template matching techniques, the pixel values of images were used. However, a reduction of the computational complexity without decreasing the image resolution will also be possible, if the images are encoded in a suitable manner (data compression!), and the correlation coefficient is calculated from the encoding coefficients. Therefore, the GPCA encoding scheme described in chapter 2 was used.

First, the GPCA primitives of the target images had to be calculated. Next, the target image, the object image, and the scaled versions of the object image were encoded by the same set of these primitives. The similarity of the object and different regions of the target image was determined by calculating the correlation coefficients of the corresponding GPCA jets.

Here, the different variances of the GPCA components have to be considered. In most cases, the ranges of the first components are significantly larger than the ranges of other components, i.e. the first components have a higher influence on the correlation coefficient and dominate its value. Thus, the GPCA components of the target image and the object images were scaled independently to obtain equal variances. The resulting component ranges were nearly identical and allowed a reasonable determination of the correlation coefficients.

The computational complexity can be varied by two parameters regulating the “blurring” of an image: The size of the subimages used to generate the GPCA jets, and the number of GPCA components. The choice of the subimage size corresponds to the choice of the image resolution at the blurred template matching. Furthermore, the number of GPCA components determines the detailed appearance of the subimages. A smaller number of components and/or larger subimages reduce the number of encoding coefficients needed to calculate the correlation coefficients and therefore the computational complexity.

The experimental results can be discussed in a few words: Compared to the blurred template matching, the objects *Stool*, *Container*, and *Chair* were recognized at the same or less computational complexity, while the objects *Container2* and *Box* were not. The calculation of the correlation coefficients by only a few components, which are most important for the reconstruction of the object images, improved the results for *Container2* and *Box*, i.e. their correlation with regions showing these objects in the target image increases, but does not become a maximum.

3.2.5 Discussion

The investigations described in this section 3.2 showed that a scale-invariant object recognition based on template matching techniques is partly successful. Furthermore, the different methods can be optimized in many ways to adapt them to specific recognition problems.

For example, the correlation coefficients could be calculated only for those pixels, which belong to the object, instead for the whole object image. In this case, the recognition of *Container2* by blurred template matching was successful, because the interfering background pixels were removed. However, this technique can not be applied to sketched objects like *Chair* or *Box*, since the pixels of these objects have identical values. Consequently, they are highly correlated with regions of uniform illumination, and thus are “recognized” at locations where the target image shows a wall or other large surfaces.

Other optimization strategies try to reduce the computational complexity. For instance, the correlation coefficient might only be calculated for regions in the target image, where the variance of the pixel values is larger than a given threshold. This prevents “empty” parts of an image without “interesting” content (like walls or other large surfaces) to be unnecessarily compared with the object image.

In spite of these additional measures, template matching techniques have fundamental drawbacks. A main point is that recognition tasks, which are invariant against scaling, rotation, etc., demand multiple searches of the object in the target image. Because of the resulting computational complexity, methods based on template matching are not suitable for fast online processing, especially if the image data is very large. For this reason, other approaches to object recognition will be investigated.

3.3 Attention-based Object Recognition

Compared to the principles of the human visual system, the strategy of object recognition techniques based on template matching appears to be rather "unintelligent". Looking at an image, humans first search for image details which attract the attention of the viewer: Such details are assumed to be more salient than others. The viewer focuses one of these salient details and recognizes them (if possible) as objects or parts of objects. Then the next detail is focused, and so on.

In contrary, methods based on template matching scan *every* detail of the whole image, regardless if they are salient or not. If necessary, this procedure is repeated for every scaling range. Thus, the object image is compared to image details even if they do not contain any interesting information and could therefore have been omitted.

For this reason, a heuristic was developed to identify the salient and most interesting regions (*points of interest*, PoI) in a given image. Image details containing points of interest are compared to the object image, whereas other details are left out. This kind of *attention-based* object recognition is expected to have a lower computational complexity than classical template matching methods.

3.3.1 Determination of Salient Regions in Images

The determination of salient image regions is a difficult task; in fact, no common definition of such regions exists. Psychological investigations show that especially regions with a high degree of symmetry attract the attention of a human viewer [LN87]. A technique based on this results is presented in [HNM96]; however, the resulting methods are rather complex and computationally expensive.

A more simple approach by Itti and Koch [IK98] characterizes the attention-attracting points of interest as the centers of image features with high intensity. To determine these centers, an image B is filtered by n different filters (feature detectors like Gabor filters, etc.). The coefficient $B^{(k)}(x, y)$ of the respective filter images $B^{(k)}$, $k = 1,\ldots, n$, are normalized to the identical range [0,1] of intensity values, weighted by a factor g_k, and added to a so-called *saliency map* S_B of the image B. The coefficients $S_B(x, y)$ of this map represent a measure for the saliency of an image pixel $B(x, y)$: The higher the value of $S_B(x, y)$, the more salient is $B(x, y)$.

Here, the weighting factors g_k are of great importance. Itti and Koch propose to set $g_k = (M_k - m_k)^2$, where M_k is the global maximum of the coefficients $B^{(k)}(x, y)$ of the k^{th} filter image $B^{(k)}$, and m_k represents the average of all local maximums of $B^{(k)}$. The idea is, that a filter output $B^{(k)}$ shall have more influence on the resulting saliency map, if it contains a few but relatively large coefficients (*peaks*). Such peaks are possible points of interest and will be emphasized in S_B. If many of the different filter images contain a peak at the same position, the corresponding coefficient in S_B will most probably represent a point of interest.

3.3.2 Point of Interest in GPCA-encoded Images

The method of Itti and Koch can be used in combination with the GPCA encoding scheme as well. Interpreting the GPCA primitives as filters, the coefficients ξ_k of the encoding jet ξ of subimage $U_{i,j}$ at position (i, j) correspond to the coefficients of the filter image $B^{(k)}$ generated by the k^{th} primitive[6]. Thus, a saliency map S_B can be generated according to the algorithm described in the previous section. However, since the coefficients $\xi_k(i, j)$ might be negative and therefore negative peaks are possible, their absolute values $|\xi_k(i, j)|$ have to be used. The generation process of S_B including the normalization and weighting of the (absolute) filter images is given by

$$S_B(i,j) = \sum_{k=1}^{n} g_k \cdot \frac{\left|\xi_k(i,j)\right|}{max_{i,j}\left(\left|\xi_k(i,j)\right|\right)} \tag{29}$$

Unfortunately, the determination of the weighting factors g_k raises a fundamental problem: In contrary to the *global* maximum, the calculation of the local maximums of an discrete, two-dimensional signal (e.g. a

[6] In the following, the notations $\xi(i, j)$ and $\xi_k(i, j)$ are used to emphasize that a jet ξ and its coefficients ξ_k belong to a certain subimage $U_{i,j}$ at position (i, j).

digital image) is very expensive in terms of the computational complexity. For this reason, an alternative method to determine the factors g_k was developed.

The weighting factors g_k will be expected to have large values, if a filter image $B^{(k)}$ contains isolated but strong peaks. Furthermore, the g_k should be small for filter images with many peaks which are nearly identical. Assuming the coefficients of a filter image (i.e. the encoding coefficients $\xi_k(i, j)$ of the GPCA) to be samples drawn from a random variable Ξ, the distribution of the $\xi_k(i, j)$ is given by the probability density function $p(\Xi)$ of Ξ. Typical measures to characterize this distribution are the *variance*, the *skewness* and the *kurtosis* (*excess*) of Ξ (see for example [PUG84]).

The higher the kurtosis $kurt(\Xi) \in [-3,\infty]$ of a random variable Ξ, the more samples of Ξ are located near the expected value $\langle\Xi\rangle$ and less samples deviate from $\langle\Xi\rangle$. In contrary, the kurtosis will become negative if many samples are scattered widely around $\langle\Xi\rangle$, and $kurt(\Xi) = 0$ if Ξ is Gaussian-distributed. Thus, Ξ is sometimes called a *super-Gaussian* ($kurt(\Xi) > 0$) or a *sub-Gaussian* ($kurt(\Xi) < 0$) random variable. Figure 31 reproduces some of the possible distributions of a Ξ with different kurtosis.

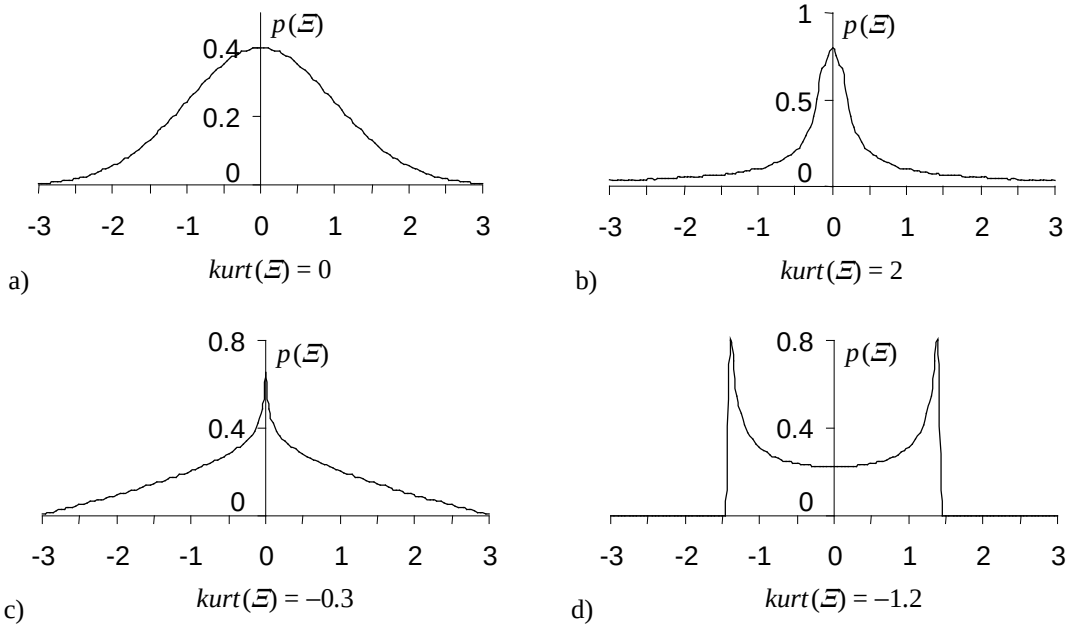


Figure 31: Examples for the probability density function $p(\Xi)$ of a random variable Ξ with different kurtosis: a) Gaussian, b) super-Gaussian, c) and d) Sub-Gaussians.

The kurtosis of the encoding coefficients $\xi_k(i, j)$ provides a measure which is similar, but not identical to the one proposed by Itti and Koch: It will deliver a high value if only a few of the $\xi_k(i, j)$ represent a strong peak. Thus, the kurtosis was chosen to calculate the weighting factors g_k:

$$g_k = kurt(\xi_k) + 3 = \frac{1}{n_x \cdot n_y} \cdot \frac{1}{var^2(\xi_k)} \cdot \sum_i \sum_j \left[\xi_k(i,j) - \overline{\xi_k}\right]^4$$

$$\text{where } \overline{\xi_k} = \frac{1}{n_x \cdot n_y} \cdot \sum_i \sum_j \xi_k(i,j) \quad \text{and} \quad var(\xi_k) = \frac{1}{n_x \cdot n_y - 1} \cdot \sum_i \sum_j \left[\xi_k(i,j) - \overline{\xi_k}\right]^2 \qquad (30)$$

Note that the value 3 is added to the kurtosis; therefore the weighting factors will never get negative.

Figure 32 shows the saliency map S_{Room} of the image *Room*. The coefficients $S_{Room}(i, j)$ are calculated according to equation (29). The number of GPCA components was $n = 16$, and the encoding coefficients $\xi_k(i, j)$ were generated from Gaussian-weighted subimages (16×16 pixels, 8 pixels overlap, $\sigma = 3$).

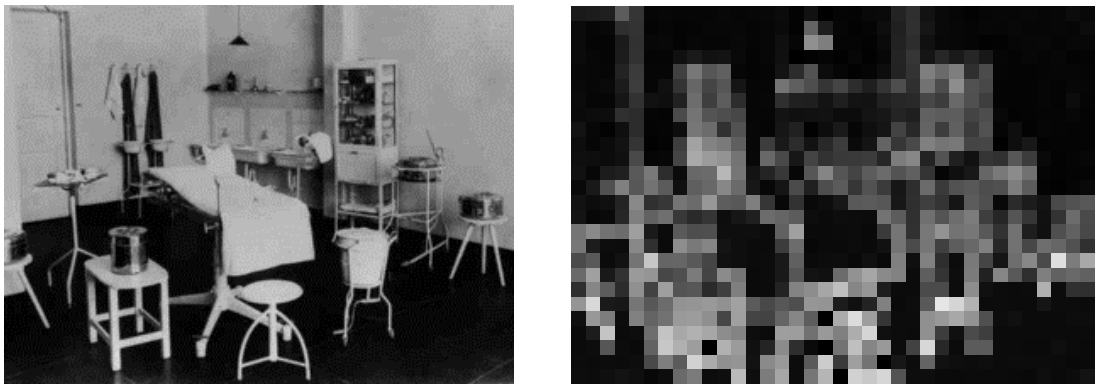


Figure 32: The image *Room* (left) and its saliency map S_{Room} (right).
Bright pixels in S_{Room} denote the points of interest.

Especially the contours of objects are marked as salient regions (bright pixels in S_{Room}), while large surfaces in *Room* with low information content (dark pixels in S_{Room}), such as walls, floor, but also the top of the operating table, are assumed to be less interesting. Here, a detailed discussion about the correspondence of the derived points of interest to their psychological motivated equivalents is omitted. However, image regions with high information content (edges, lines, etc.) are detected, which is an important image processing task.

3.4 Development of a Scale-invariant Model for Objects

The technique to detect the points of interest as described in the previous section can be applied to object images as well. Possibly, only those encoding jets representing the most salient image regions are needed to recognize the object: Since the other jets encode "unimportant" parts of the object image, they can be omitted.

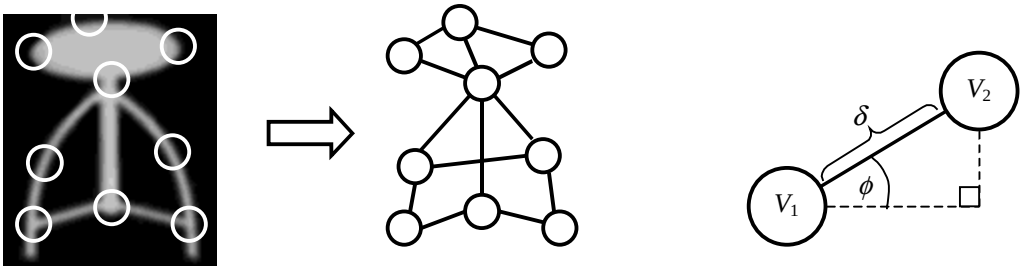
This idea leads to the development of an object model consisting only of the jets ξ of salient points. Ideally, this model will be invariant to the size of the represented object, if the *relative* positions of the jets alone are stored. However, because of the limited invariance of the jets against scaling, the model is expected to be limited scale-invariant as well.

The investigations concerning this promising approach are still in progress and will be subject to future research within the scope of the project SEMACODE. Comparable methods were used by others to implement a successful system for face recognition purposes [WFK99].

3.4.1 Object Modeling Using Points of Interest

The idea of modeling an object by its points of interest will be demonstrated for the simple example of the object image *Stool*. Figure 33 shows the object image with typical points of interest and the resulting model. For instance, this model can be represented by a *labeled graph*: Each node in the graph describes a point of interest and therefore holds the corresponding encoding jet, whereas each edge is labeled with the relative position of the points at its two nodes.

Principally, this object model is invariant against the sizing of the object: Given the angle ϕ and the distance δ between two nodes, the relative position of the nodes can be represented by a tuple (ϕ, δ) of polar coordinates in *dimensionless units*. In case of the angle ϕ, this true by definition; the distance δ can be calculated from the "real" distance of the nodes divided by the average distance of all nodes. Obviously, this *relative distance* is independent of the size of the object. Furthermore, *rotated* versions of the object are easily represented by the model: The angle of rotation ϕ_{rot} alone has to be added to the angle ϕ of each edge in the model graph.



object image with points of interest

model graph of the object

representation of the relative position of V_1 and V_2 in polar coordinates (ϕ, δ)

Figure 33: Typical points of interest of the object image *Stool* (left) and the resulting model graph (middle). The label of the edge (V_1 , V_2) represents the relative position of two nodes V_1 and V_2 in polar coordinates (ϕ, δ) (right).

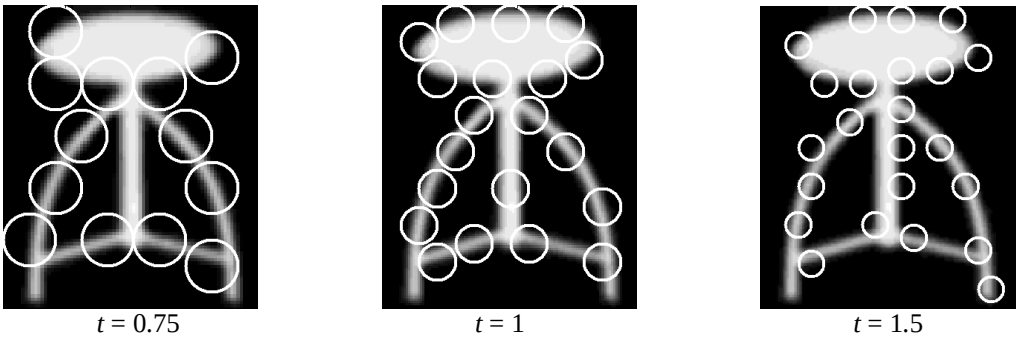
Using the GPCA image encoding technique, several versions of the object image *Stool* differing in their size were transformed into corresponding object models (*multi-resolution object model*). First, the points of interest had to be calculated according to section 3.3.2. The nodes of the model graphs resulted from those points with coefficients $S_B(i, j)$ of the saliency map S_B, which were greater or equal than the average $\langle S_B(i, j)\rangle$.

However, many of these coefficients were located nearby and more or less identical. To achieve a distribution similar to Figure 33, where the points of interest are located as widely as possible, and where the number of points remains almost constant at each scaling range, only coefficients $S_B(i, j)$ and $S_B(i', j')$ with a *distance*

$$\Delta[S_B(i, j), S_B(i', j')] = \sqrt{(i - i') + (j - j')} \qquad (31)$$

at least greater or equal than the *minimum dynamical distance* $d_t = t \cdot d_1$ were selected as points of interest. Here, d_1 is the smallest distance between two points of interest, which is allowed at the original scale $t = 1$, and has to be chosen empirically.

For $d_1 = 2.5$, Figure 34 shows the selected points of interest of the object image *Stool* at different scaling ranges. The points of interest are marked by circles representing the support of the underlying subimage; the subimages were 16×16 pixels in size (8 pixels overlap), and the number of encoding coefficients $\xi_k(i, j)$ per jet $\xi(i, j)$ was 16.



$t = 0.75$ $t = 1$ $t = 1.5$

Figure 34: Selected points of interest of *Stool* at different scaling ranges. Circles represent the support of the underlying subimages $U_{i,j}$ (16×16 pixels, 8 pixels overlap). The corresponding coefficients $S_B(i, j)$ were greater or equal than the average $\langle S_B(i, j)\rangle$, and their distance was at least d_t pixels ($d_1 = 2.5$).

Each node generated from the selected points was labeled with the encoding jet $\xi(i, j)$ of its corresponding subimage, while the attributes of the edges were set to the relative node positions. Unfortunately, as mentioned above, the scale invariance of the resulting object model is limited to a small range, since the

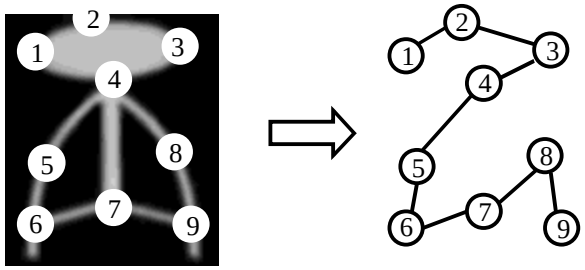
encoding jets $\xi(i, j)$ remain constant only for slight changes in the size of the object image. For this reason a separate object model was calculated at several scaling ranges.

3.4.2 Using the Model for Recognizing Objects

In analogy to the investigations in section 3.2, the developed object model was used to recognize several objects in the target image *Room*. For efficiency reasons, minor modifications had to be applied to the representation of the model graph.

For example, some edges of the graph in Figure 33 can be omitted: There exists a graph with a smaller number of edges, which represents the relative position of all the nodes, i.e. where at least one edge is attached to each node. A graph with the smallest possible number of edges will result, if at least one and at most two edges are attached to each node: This graph is called a *path graph* or simply *path*[7], since it resembles a path in the fully connected graph.

In case of the actual object model, the construction of a path is rather simple. Starting with a point of maximum saliency (i.e. where the corresponding coefficient $S_B(i, j)$ is the maximum of the saliency map S_B), the *fixation point*, which is transformed into the first node of the graph, the next point of interest with a minimum distance $d_t = t \cdot d_1$ and the highest remaining coefficient $S_B(i', j')$ is searched. The new point is transformed into a node as well and connected to the previous node by an edge representing their relative position. This procedure is repeated until no more point of interest satisfying the above conditions are left. Figure 35 shows a simplified example path for the object *Stool* (the real path graph of *Stool* is much more complex since there are more point of interest in the object image).



points of interest resulting path

Figure 35: Generation of a path by nine coefficients $S_B(i, j)$ ordered according to descending values.

After the generation of the path, the points of interest of the target image *Room* were determined. Of these points, the point *A* with an associated encoding jet most similar to the encoding jet of the fixation point of *Stool* was selected. Therefore the correlation coefficient of both jets had to be calculated. Here, the differing variances of the encoding coefficients $\xi_k(i, j)$ were equalized by normalizing the components of the jets according to section 3.2.4.

Beginning with the fixation point of *Stool*, the point *P* in the target image corresponding to the next node in the model path was searched. The search could be reduced to a small part of the target, since the relative position (ϕ, δ) in reference to the fixation point was given by the label of the connecting edge (see Figure 36). This part resulted from a region where the inclination of the connecting line *AP* between the first point *A* and the next point *P* differed at most about $\pm$ 7,5° from the predicted angle ϕ. Furthermore, *AP* must not be shorter than one half and not longer than two times the length of the *absolute* distance between the corresponding points in the object model. These restrictions arise from the following two observations:

- The object (possibly) appearing in the target image may be slightly rotated as compared to the object given in the object image. Such a rotation is likely to result from the discrete pixel structure of digital images and has to be compensated, even if the recognition task is not expected to be invariant against rotation.

[7] Note that a path graph is not necessarily the only graph with a minimum number of edges.

- So far, the invariance of the object model against scaling is limited (see previous section). Thus, the size of the searched object appearing in the target image (and with it the length of *AP*) has to be within a limited scaling range.

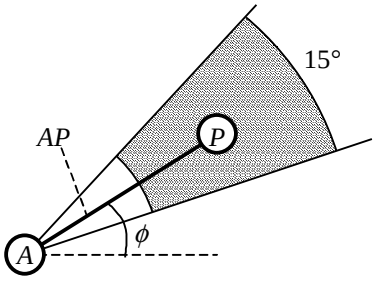


Figure 36: Possible target image locations of the point *P* in reference to the point *A* (shaded).

A successful detection of the point *P* in the target image allowed a first approximation of the size and the rotation of the object, which was used to improve the search for further points according to the technique described above. If *P* could not be detected, subsequent points of the model path were searched, or else the search task restarted at the next point of the target image, which was most similar to the fixation point of the object image.

This method is more complex but faster and more robust than the rather simple blurred template matching. Consequently, the experimental results included the successful recognition of the objects *Stool*, *Container*, *Chair*, and even *Container2*. However, the object *Box* could not be recognized as well.

3.5 Discussion and Further Research

The development of an automated object recognition system resulted in an object model with a limited invariance against scaling, which is based on the proper selection of salient image regions (points of interest). Furthermore, the model is easily expanded to support a rotation-invariant object representation. By utilizing points of interest, the robustness of the recognition process against several distortion effects such as occlusion is enhanced.

Further research in the scope of the project SEMACODE will concentrate on the improvement of the object model, especially its invariance against scaling. Additionally, the investigation of other similarity measures is considered.

Besides, a general object recognition model shall be developed on the basis of the results and methods described herein. For instance, latest neuro-biological and psychological findings should be considered to improve the recognition task. Here, the main subject will be the investigation of Gestalt-theoretic principles, which are only scarcely used in image processing.

Appendix: Efficient Scaling of Images

Nearly all the methods described in chapter 3 require an image B, which is represented by pixel values $B(x, y)$, to be resized as fast and correct as possible. Trivial solutions, such as leaving out or copying pixels in regular intervals, are too inaccurate and not acceptable. Instead, an efficient technique based on signal-theoretic results was developed.

To simplify the investigations, images are first assumed to be one-dimensional signal. The case of "real" images, i.e. two-dimensional signals, will be considered later. The amount of resizing (scaling) is denoted by the *scaling factor t* calculated from the quotient of the width/height of the scaled image B_t and the width/height of the original image B. Note that in this case the area of B_t is t^2 times the area of B.

Decimation and Expansion

The most simple operations to scale a finite, one-dimensional signal B consisting of N samples $B(x)$, where $x \in \{0,\ldots,N-1\}$, are *decimation* and the *expansion*. They allow the downsizing of B by a factor M and the upsizing of B by a factor L, respectively. Here, both M and L have to be real integer numbers. The mathematical definitions of decimation and expansion are (according to [VAI93])

$$\text{M-fold decimation:} \quad B_D(x) = B(x \cdot M) \quad \text{if } x \cdot M \in \{0,\ldots,N-1\}$$

$$\text{L-fold expansion:} \quad B_E(x) = \begin{cases} x/L & \text{if } x \bmod L = 0 \text{ and } x/L \in \{0,\ldots,N-1\} \\ 0 & \text{else} \end{cases} \tag{32}$$

The decimated signal B_D consists of each M^{th} sample of B, whereas the expanded signal B_E results from inserting $L-1$ zero samples between each pair of successive samples of B. Thus, B_D is downsized by a factor $1/M$ and B_E is upsized by a factor L compared to the original signal B. Figure 37 shows the results of applying decimation and expansion to a simple signal.

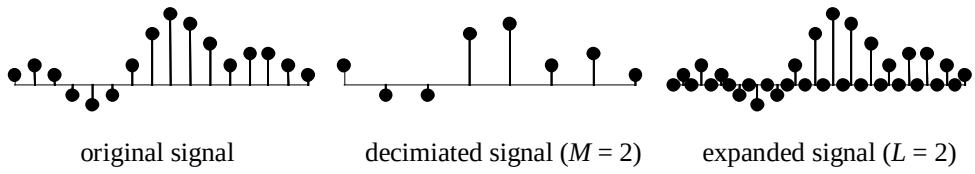


Figure 37: Decimation and expansion of a signal.

However, neither B_D nor B_E are suitable approximations of a signal B to be scaled by a factor $t = 1/M$ or $t = L$, respectively: Previous to the decimation, the signal B has to be filtered by a decimation filter H_D to avoid the effect of aliasing (see below). Likewise, the expanded signal B_E must be smoothed by an interpolation filter H_E to fill the "gaps" resulting from inserted zero samples.

The appearance of frequency components in a decimated signal, which have no equivalent components in the original signal, is called *aliasing*. For example, let B_ν be a signal with N_ν samples $B_\nu(x)$ where $B_\nu(x) = 1$ if x is a multiple of 2, and $B_\nu(x) = 0$ else. Obviously, B_ν is periodical with the smallest possible period $T_\nu = 2$ for integer coordinates x. In signal theory, the *frequency* ω of a signal is usually given by $\omega = 2 \cdot \pi / T$, where T denotes the period of the signal. Thus, the frequency ω_ν of the signal B_ν is $\omega_\nu = 2 \cdot \pi / T_\nu = \pi$. Since T_ν can not get smaller than 2, $\omega_\nu = \pi$ represents the *highest possible* frequency, which is called the *Nyquist frequency*.

Now, let $B_{\nu, D}$ be the signal resulting from the M-fold decimation of B_ν where $M = 3$. The samples of $B_{\nu, D}$ are given by $B_{\nu, D}(x) = 1$ if x is a multiple of 2, and $B_{\nu, D}(x) = 0$ else. Thus, $B_{\nu, D}$ is identical to B_ν but contains only one third of the samples of B_ν. Consequently, its frequency $\omega_{\nu, D} = \pi$ is equal to ω_ν as well. However, since $B_{\nu, D}$ is a version of B_ν resized by a scaling factor $t = 1/M = 1/3$, its period $T_{\nu, D}$ should be one third of the original period T_ν resulting in a theoretical frequency of $\omega_{\nu, D} = 3 \cdot \pi$. But this theoretical

frequency is higher that the Nyquist frequency which is impossible. Therefore $B_{v,D}$ is not a valid approximation of the M-fold decimated signal B_v.

It can be shown that each frequency component higher than π/M may lead to aliasing in a M-fold decimated signal [VAI93]. For this reason, such frequency components are removed before decimating a signal. This is done by applying a *decimation filter* H_D with a *cutoff frequency* $\omega_D = \pi/M$. H_D is a so-called *lowpass filter*: Ideally, the output signal of H_D contains only those frequency components which are equal or lower than the cutoff frequency.

In the case of a L-fold expansion, the expanded signal has to be smoothed. This is done by applying a lowpass filter, the *interpolation filter* H_E, with cutoff frequency $\omega_E = \pi/L$ after the expansion stage.

Scaling Filters

Decimation and expansion operations can be combined with their corresponding filters H_D and H_E respectively. For mathematical reasons, the signal to be processed and the filters consisting of the filter coefficients $H_D(i)$ and $H_E(i)$ are expected to have an infinite support. This is done by "embedding" a finite signal B with N samples $B(x)$, where $x \in \{0,\ldots,N-1\}$, into an infinite number of zero samples, i.e. $B(x) = 0$ if $x \notin \{0,\ldots,N-1\}$; the same holds for finite filters. According to [VAI93], the improved M-fold decimation and L-fold expansion operations are given by

$$\begin{aligned} &\textit{M-fold decimation}: && B_D(x) = \sum_{i=-\infty}^{\infty} B(x\cdot M - i)\cdot H_D(i) = \sum_{i=-\infty}^{\infty} B(i)\cdot H_D(x\cdot M - i) \\ &\textit{L-fold expansion}: && B_E(x) = \sum_{i=-\infty}^{\infty} B(i)\cdot H_E(x - i\cdot L) \end{aligned} \qquad (33)$$

Applying a M-fold decimation operation to the output of a L-fold expansion operation, the signal B can be resized by any (rational) scaling factor $t = L/M$. With equation (33), the samples of the scaled signal B_t are

$$\begin{aligned} B_t(x) &= \sum_{j=-\infty}^{\infty} H_D(j)\cdot \sum_{i=-\infty}^{\infty} B(i)\cdot H_E(x\cdot M - i\cdot L - j) \\ &= \sum_{i=-\infty}^{\infty} B(i)\cdot \sum_{j=-\infty}^{\infty} H_D(j)\cdot H_E(x\cdot M - i\cdot L - j) \end{aligned} \qquad (34)$$

The second row of equation (34) represents a combined decimation and expansion where the filter coefficients are given by convolutions of H_D and H_E: These coefficients generate a "new" filter H_F. Since H_D and H_E are lowpass filters, H_F is a lowpass filter with cutoff frequency[8] $\omega_F \approx min(\omega_D, \omega_E) = \pi/max(M, L)$. Thus, the convolution summation over j in equation (34) can be replaced by the coefficients of the *scaling filter* H_F to obtain the classical one-dimensional *t-fold scaling operation* of a signal B ($t = L/M$):

$$B_t(x) = \sum_{i=-\infty}^{\infty} B(i)\cdot H_F(x\cdot M - i\cdot L) \qquad \text{where} \qquad H_F(k) = \sum_{j=-\infty}^{\infty} H_D(j)\cdot H_E(k - j) \qquad (35)$$

Efficient Scaling Filters

Considering that the coefficients $B(k)$ are zero where $k \notin \{0,\ldots,N-1\}$, the summation in equation (35) has to be computed for at most N indices $i \in \{0,\ldots,N-1\}$. Usually, the scaling filter H_F has a finite number of non-zero filter coefficients which is significantly smaller than N. Thus, many of the filter coefficients $H_F(x\cdot M - i\cdot L)$ are zero as well, and the number of summands in equation (35) is expected to be less than N.

[8] The cutoff frequency of H_F is $\omega_F = min(\omega_D, \omega_E)$ for ideal lowpass filters H_D and H_E. However, real lowpass filters might damp frequencies lower than the cutoff frequency and do not fully reject higher frequencies: Depending on the used filter implementations, ω_F will slightly differ from $min(\omega_D, \omega_E)$.

For the purpose of image scaling, the filter H_F is chosen to be symmetrical to the origin. This ensures a uniform filter operation without phase shifting (i.e. the signal is not shifted or "delayed" compared to the origin $x = 0$). Consequently, the filter coefficients satisfy the properties $H_F(k) = H_F(-k)$ where $k \in \{-K,\ldots,K\}$ and $H_F(k) = 0$ if $k \notin \{-K,\ldots,K\}$. K is called the *radius* of the scaling filter; the total number of filter coefficients is $2 \cdot K+1$.

Following the above considerations, each summand in equation (35) with an associated index i satisfying neither $i \in \{0,\ldots,N-1\}$ nor $-K \le x \cdot M - i \le K$ has to be zero and can be omitted. Thus, equation (35) is equivalent to

$$B_t(x) = \sum_{i=start(x,M,L,K)}^{stop(x,M,L,K,N)} B(i) \cdot H_F(x \cdot M - i \cdot L)$$

$$\text{where} \quad start(x, M, L, K) = max\left(0, \left\lceil \frac{x \cdot M - K}{L} \right\rceil\right)$$

$$\text{and} \quad stop(x, M, L, K, N) = min\left(\left\lfloor \frac{x \cdot M + K}{L} \right\rfloor, N - 1\right) \qquad (36)$$

Scaling 2-dimensional Signals

Given a two-dimensional signal B with $N_x \times N_y$ samples (i.e. a "real" image which is $N_x \times N_y$ pixels in size) and a two-dimensional scaling filter H_{2F} with $(2 \cdot K-1) \times (2 \cdot K-1)$ coefficients, equation (36) may be formulated in the two-dimensional case:

$$B_t(x, y) = \sum_{i=start(x,M,L,K)}^{stop(x,M,L,K,N)} \sum_{j=start(y,M,L,K)}^{stop(y,M,L,K,N)} B(i, j) \cdot H_{2F}(x \cdot M - i \cdot L, y \cdot M - j \cdot L) \qquad (37)$$

Equation (37) will be simplified, if H_{2F} is separable, i.e. H_{2F} can be written as the product of two one-dimensional filters. Given that $H_{2F}(x, y) = H_F(x) \cdot H_F(y)$, equation (38) holds:

$$B_t(x, y) = \sum_{i=start(x,M,L,K)}^{stop(x,M,L,K,N)} H_F(x \cdot M - i \cdot L) \cdot \sum_{j=start(y,M,L,K)}^{stop(y,M,L,K,N)} B(i, j) \cdot H_F(y \cdot M - j \cdot L) \qquad (38)$$

First, the image columns are resized by an one-dimensional scaling operation. Next, the same is done with the resulting, vertically scaled rows.

Choice of Filter Coefficients

As mentioned above, H_F should be a separable lowpass filter which is symmetrical to the origin. For image processing purposes, the two-dimensional *Gaussian filter* G_σ is a suitable choice:

$$H_{2F}(x, y) := G_\sigma(x) \cdot G_\sigma(y) \quad \text{where} \quad G_\sigma(k) = \frac{1}{\sigma \cdot \sqrt{2 \cdot \pi}} \cdot exp\left(-\frac{k^2}{2 \cdot \sigma^2}\right), \; k \in \{-K,\ldots,K\} \qquad (39)$$

The frequency response of the Gaussian filter G_σ is given by its Fourier transform

$$\hat{H}(\omega_x, \omega_y) = \hat{G}_\sigma(\omega_x) \cdot \hat{G}_\sigma(\omega_y) \qquad \text{where} \qquad \hat{G}_\sigma(\omega) = exp(-\tfrac{1}{2} \cdot \omega^2 \cdot \sigma^2) \qquad (40)$$

determined by the parameter σ [JÄH89]. Figure 38 reproduces this frequency response as a function of σ.

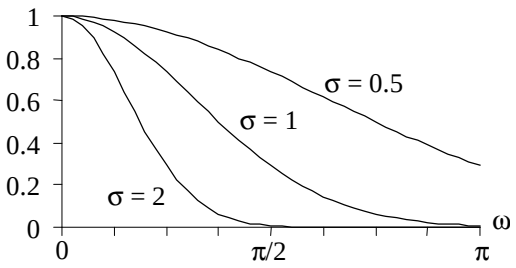


Figure 38: Frequency response of a one-dimensional Gaussian filter G_σ as a function of σ.

G_σ is not an ideal lowpass filter, since frequency components of the signal B higher than the cutoff frequency $\omega_F = \pi / max(M, L)$ are not fully rejected and lower frequency components are slightly damped. Therefore, an additional parameter d has to be given to obtain a suitable choice of σ. Here, d represents the minimum factor used to attenuate frequency components higher than the cutoff frequency ω_F. With $\omega_F \in [0, \pi]$, the parameter σ is calculated by

$$\hat{G}_\sigma(\omega_F) = exp(-\tfrac{1}{2} \cdot \omega_F^2 \cdot \sigma^2) \le d \quad \Rightarrow \quad \sigma \ge \sqrt{-2 \cdot ln(d)} \cdot max(M, L) / \pi \qquad (41)$$

Applied to most real-life images, a parameter value of $d = 0.5$ is known to produce acceptable results. To determine the number $2 \cdot K + 1$ of filter coefficients, a radius K, which is three times the value of σ, should be sufficient:

$$K = \lceil 3 \cdot \sigma \rceil \qquad (42)$$

References

[ARL97] B. Arlt: *Verfahren zur Bildkodierung mit künstlichen neuronalen Netzen*, Diploma Thesis, Fachbereich Informatik (FB20), J. W. Goethe-Universität Frankfurt/Main, 1997

[AB98a] B. Arlt, R. Brause: *The Principal Independent Components of Images*, Internal Report 1/98, Fachbereich Informatik (FB20), J. W. Goethe-Universität Frankfurt/Main, 1998 http://www.cs.uni-frankfurt.de/asa/

[AB98b] B. Arlt, R. Brause: *The Principal Independent Components of Images*, in: L. F. Niklasson, M. B. Boden, T. Ziemke (eds.): *Proc. ICANN'98*, Vol.2, Sweden, Springer-Verlag, 1998

[AB98c] B. Arlt, R. Brause: *Image Encoding with Principal Independent Components*, in: O. Herzog, A. Günter: *Proc. Künstliche Intelligenz* (*KI-98*), LNCS 1504, Springer Verlag 1998

[BS96] A. J. Bell, T. J. Sejnowski: *Edges are the 'Independent Components' of Natural Scenes*, Advances in Neural Information Processing Systems 9, MIT Press, 1996

[BRA95] R. Brause: *Neuronale Netze*, Teubner-Verlag Stuttgart, 1995

[BRE93] T. M. Breuel: *View-Based Recognition*, IDIAP Research Report 93-09, 1993

[COM94] P.Comon: *Independent Component Analysis – A New Concept?*, Signal Processing 36/1994, pp. 287-314, 1994

[DR87] W. B. Davenport, W. L. Root: *An Introduction to the Theory of Random Signals and Noise*, IEEE Press New York, NY, 1987

[FIE87] D. J. Field: *Relations Between the Statistics of Natural Images and the Response Properties of Cortical Cells*, Journal of the Optical Society of America, Vol.4, No.12, pp.2379-2393, 1987

[GAB46] D. Gabor: *Theory of Communication*, Journal of the IEE, Vol.93, pp. 429-457, 1946

[GRE96] H. Greenspan: *Non-Parametric Texture Learning*, in: S. K. Nayar, T. Poggio (Eds.): *Early Visual Learning*, pp. 299-328, Oxford University Press, New York, 1996

[GW93] R. C. Gonzalez, R. E. Woods: *Digital Image Processing*, Addison-Wesley, 1993

[HNM96] G. Heidemann, T. Nattkemper, G. Menkhaus, H. Ritter: *Blicksteuerung durch präattentive Fokussierungspunkte*, contribution to the workshop *Aktives Sehen in technischen und biologischen Systemen* der GI-Fachgruppe 1.0.4 Bildverstehen, Hamburg, 1996

[HO97] A. Hyvärinen, E. Oja: *A Fixed-Point Algorithm for Independent Component Analysis*, Neural Computation 9, pp. 1483-1492, 1997

[HPB98] T. Hofmann, J. Puzicha, J. Buhmann: *Unsupervised Texture Segmentation in a Deterministic Annealing Framework*, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol.20, No.8, 1998

[HW71] A. Habibi, P. A. Wintz: *Image Coding by Linear Transformation and Block Quantization*, IEEE Transactions on Communication Technology, Vol. COM-19, No.1, pp. 50-62, 1971

[IK98] L. Itti, C. Koch: *A Model of Saliency-Based Visual Attention for Rapid Scene Analysis*, IEEE Trans. Pattern Analysis and Machine Intelligence (PAMI), 20(11), pp.1254-1259, 1998

[JÄH89] B. Jähne: *Digitale Bildverarbeitung*, Springer-Verlag Berlin, 1989

[KK92] K. D. Kammeyer, K. Kroschel: *Digitale Signalverarbeitung – Filterung und Spektralanalyse*, Teubner-Verlag Stuttgart, 1992

[LIN94] T. Lindeberg: *Scale-Space Theory in Computer Vision*, Kluwer Academic Publishers, Dordrecht, Niederlande, 1994

[LN87] P. Locher, C. Nodine: *Symmetry Catches the Eye*, in: A. Levy-Schoen, J. O'Reagan (Eds.): *Eye Movements: From Physiology to Cognition*, pp.353-361, Elsevier Science Publishers B.V., Niederlande, 1996

[LVH94] N. K. Logothetis, T. Vetter, A. Hurlbert, T. Poggio: *View-Based Models of 3D Object Recognition and Class-specific Invariances*, A.I.Memo 1472, MIT AI Laboratory & Center for Biol. and Comp. Learning, 1994

[OF96] B. A. Olshausen, D. J. Field: *Natural Image Statistics and Efficient Coding*, Network: Computation in Neural Systems, Vol.7, pp. 333-339, 1996

[OLS96] B. A. Olshausen: *Learning Linear, Sparse, Factorial Codes*, A.I. Memo 1580 (C.B.C.L. Paper 138), MIT AI Laboratory, Center for Biol. & Comp. Learning, Dept. of Brain and Cognitive Science, 1996

[PRA78] W. K. Pratt: *Digital Image Processing*, John Wiley & Sons, New York, 1978

[PT85] T. Poggio, V. Torre, C. Koch: *Computational Vision and Regularization Theory*, Nature 317, pp. 314-319, London, 1985

[PUG84] V. S. Pugacev: *Probability Theory and Mathematical Statistics for Engineers*, Pergamon Press, Oxford, 1984

[PZ89] M. Porat, Y.Y. Zeevi: *Localized Texture Processing in Vision: Analysis and Synthesis in the Gaborian Space*, IEEE Transactions on Biomedical Engineering, Vol.36, No.1, pp. 115-129, 1989

[QBC95] M. Flickner et al.: *Query by Image and Video Content: The QBIC System*, IEEE Computer, September 1995, pp.23-32

[SAN89] T. D. Sanger: *Optimal Unsupervised Learning in a Single-Layer Linear Feedforward Neural Network*, Neural Networks, Vol.2, pp.459-473, 1989

[SJ99] S. Santini, R. Jain: *Similarity Measures*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 1999 (in print)
http://vision.ucsd.edu/~ssantini/articles/vision/SimPaper.ps.Z

[VAI93] P. P. Vaidyanathan: *Multirate Systems and Filter Banks*, Prentice Hall PTR, Englewood Cliffs, New Jersey, 1993

[WFK99] L. Wiskott, J. M. Fellous, N. Krüger, C. von der Malsburg: *Face Recognition by Elastic Bunch Graph Matching*, in L.C. Jain et al. (Eds.): *Intelligent Biometric Techniques in Fingerprint and Face Recognition*, Springer-Verlag, 1999