

Independent component analysis for texture segmentation

R. Jenssen¹, T. Eltoft

Department of Physics, University of Tromsø, N-9037 Tromsø, Norway

Abstract

Independent component analysis (ICA) of textured images is presented as a computational technique for creating a new data dependent filter bank for use in texture segmentation. We show that the ICA filters are able to capture the inherent properties of textured images. The new filters are similar to Gabor filters, but seem to be richer in the sense that their frequency responses may be more complex. These properties enable us to use the ICA filter bank to create energy features for effective texture segmentation. Our experiments using multi-textured images show that the ICA filter bank yields similar or better segmentation results than the Gabor filter bank.

Key words: Independent component analysis; Image model; Data dependent filter bank; Texture segmentation; Energy features

1 Introduction

Automatic segmentation of multi-textured images is an important area of image processing which has received considerable attention during the last decade. In this paper we introduce *independent component analysis* (ICA) [1–4] of textured images as a computational technique for creating a new *data dependent* filter bank for use in texture segmentation.

Independent component analysis is a generative model for observed multivariate data, which are assumed to be mixtures of some unknown latent variables. Originally, it emerged as a means to solve the problem of blind source separation (BSS) [3,5,6]. BSS refers to the problem of finding the original source signals from available mixtures, without any prior knowledge of the number of sources or the mixing

¹ Corresponding author. Tel. (+47) 776 45184, Fax. (+47) 776 45580,
Email: robertj@phys.uit.no

mechanisms. In the linear case, the observed signals $\mathbf{x}(t)$ can be expressed as:

$$\mathbf{x}(t) = \mathbf{A}\mathbf{s}(t), \quad (1)$$

where $\mathbf{A}$ is called the mixing matrix and $\mathbf{s}(t)$ is the source signals.

Based on the assumption that the source signals are mutually statistically independent, the solution is obtained in an unsupervised learning process, which finds the de-mixing matrix $\mathbf{W}$, such that $\mathbf{y}(t) = \mathbf{W}\mathbf{x}(t)$ is an estimate of $\mathbf{s}(t)$. The components of $\mathbf{y}(t)$ are required to be as mutually independent as possible.

The model in equation (1) is referred to as the basic ICA model [4]. Thus, in blind source separation it is the source signals which are the “independent components”. This model has in the recent years been widely applied in diverse areas like biomedical imaging [7–10], telecommunications [11,12], time series prediction [13–15], audio separation [16] and seismic monitoring [17]. See e.g. [18–20] for more references on ICA applications.

ICA has also been proposed as a generic statistical model for images [21–25]. In this case an image $\mathbf{x}$ is modeled as:

$$\mathbf{x} = \sum_{i=1}^N s_i \mathbf{a}_i, \quad (2)$$

where $\mathbf{a}_i$, $i = 1, \dots, N$, are referred to as ICA image basis functions, and s_i , $i = 1, \dots, N$, are statistically independent weighting components. It also turns out that the s_i -components exhibit so-called sparseness, i.e. only a few of the weighting coefficients will have a value significantly deviating from zero. This property is exploited when the model is used in threshold-based image denoising (sparse code shrinkage) [26,27]. Equation (2) was in [21] originally motivated as a model for the neurons in the primary sensory areas of the brain.

In this paper we propose to use the ICA image model of equation (2) to generate a *data dependent filter bank for texture segmentation*. The filter bank consists of the ICA basis images, learned from textured images. We show that these basis images are able to capture the inherent structure of the texture, and hence enable us to create features for effective texture segmentation of composite input textures.

Traditionally, texture segmentation techniques are classified as *statistical methods*, *model based methods* or *filtering methods*.

The statistical methods are often based on the co-occurrence matrix [28], which gives information about the second-order statistics of a textured image. Texture features which can be extracted from the co-occurrence matrix, are based on parameters quantifying e.g. energy, entropy, contrast, homogeneity or correlation [29–31].

In model based methods a textured image is modeled as a probability model, and

the coefficients of the model are used to characterize the images [32]. These models include Gaussian Markov random fields [33], Gibbs random fields [34] and Wold models [35,36].

Our ICA based texture segmentation method belongs to the filtering methods, also called signal processing methods [37], which include a whole family of algorithms. This approach also found motivation in psycho-physiology. Studies by e.g. Campbell and Robson [38] and De Valois *et al.* [39] suggested that the brain performs a multi-channel frequency and orientation analysis of the visual image formed on the retina [37]. The basic idea in the filtering methods is that a composite textured image is filtered through a bank of filters, and features appropriate for texture segmentation are generated based on the filter outputs [40].

Laws' texture energy filters [41,42] represent an early example of this approach. Law's generated 2-D filter kernels designed to detect "spots", "edges" and "rings". The filters are convolved with a composite textured image to produce a set of texture energy images. Each pixel in the input image is hence represented by a vector of energy features. Subsequent methods often follow this outline, but using different filter banks. For example, Coggins and Jain [43] suggested using filters characterized in the spatial frequency domain as ring filters with widths between one - and two octaves, and with center frequencies one octave apart. The choice of filter bandwidth and center frequency separation were motivated from experiments showing that the frequency bandwidth of simple cells in the striate cortex is about one octave [44]. Wedge-shaped orientation channels were also implemented, with directions along the horizontal, vertical and the two diagonals.

Marcelja [45] and Daugman [46] showed that 2-D Gabor filters are good models of the receptive fields in the striate cortex. In the last couple of decades Gabor filters have been widely used in texture segmentation [47–51]. Gabor filters can be interpreted as oriented band-pass filters [49].

Jain and Farrokhnia [49] used a dyadic even-symmetric Gabor filter bank to generate energy features for texture segmentation, and reported successful results. Their filters are oriented along the horizontal, vertical and the two diagonals, and the number of center frequencies, i.e. the size of the filter bank, varies according to the size of the textured image. We will use Jain and Farrokhnia's method for comparison with ICA based texture segmentation.

Wavelet and wavelet packets methods [52–57] of various sub-band decompositions have also been successfully applied in texture analysis. With an appropriately chosen basis set, the 2-D wavelet transform offers spatial frequency and orientation selectivity, and texture features at the resolution levels containing most of the energy activity can be obtained [29,52]. The Gabor filters form an approximate wavelet basis, with the Gabor function as the wavelet [49].

The ICA based approach is different from existing filtering methods in that it produces a data dependent filter bank. The filters are obtained from training data using an unsupervised algorithm, and the user is hence required to provide training data

containing the appropriate structure. Such data may be found in available image or texture libraries.

In the next section we present in some detail independent component analysis as a generic statistical model for images. Next, in section 3 we specifically discuss the properties of the ICA filter bank obtained from a training set of texture data. We also briefly review the Gabor filter theory. In section 4 we explain how the ICA filter bank method can be used for feature generation, and in section 5 we present some texture segmentation experiments where we compare the performance of the ICA filter bank and the Gabor filter bank. In section 6, we make some concluding remarks.

Finally, for the convenience of the reader unfamiliar with ICA, we include in appendix A a brief review of some basic ICA theory.

2 Independent component analysis of images

[Fig. 1 about here.]

Independent component analysis of images produces a set of basis functions [21,25,24]. Hence, as noted from equation (2), an image may be expressed as a weighted sum of these basis functions, with weighting components s_i , $i = 1, \dots, N$. The basic idea in the ICA image model is to construct basis functions which for a given image result in weighting components which are mutually statistically independent. Thus, the s_i 's in equation (2) are the "independent components". The image model described here is illustrated in figure 1.

In the sequel, we represent an image as a column vector $\mathbf{x} = [x_1, \dots, x_M]^T$ by re-shaping the image matrix row-by-row into a single column. In this way we avoid using four-dimensional matrices in the ICA filtering, and an image may thus be represented using the standard ICA methodology. Mathematically, we express the model as,

$$\mathbf{x} = \sum_{i=1}^N s_i \mathbf{a}_i = \mathbf{A} \mathbf{s}, \quad (3)$$

where the basis functions $\mathbf{a}_i$, $i = 1, \dots, N$, are the columns of the $(M \times N)$ matrix $\mathbf{A}$, and $\mathbf{s} = [s_1, \dots, s_N]^T$.

In order to learn the mixing matrix, $\mathbf{A}$, we attempt to find a linear transformation $\mathbf{W}$ of the data $\mathbf{x}$, which yields a vector $\mathbf{y}$, estimating $\mathbf{s}$, whose components are as statistically independent as possible. Accordingly,

$$\mathbf{y} = \mathbf{W} \mathbf{x}. \quad (4)$$

If $N = M$, $\mathbf{W}$ is the inverse of $\mathbf{A}$, i.e. $\mathbf{W} = \mathbf{A}^{-1}$. There exists no analytical solution to this problem.

In some cases it is expedient to choose $N < M$, i.e. the dimension of $\mathbf{y}$ is less than the dimension of $\mathbf{x}$. Dimensionality reduction can be accomplished by principal component analysis (PCA). PCA may be considered a preprocessing step towards statistical independence, since it removes second order correlations in the training data. This is often called whitening. The PCA preprocessing and dimensionality reduction is discussed in appendix A.

Since the weighting components in the ICA model are mutually statistically independent, one might say that ICA attempts to produce a code $\mathbf{y}$ of $\mathbf{x}$, where the joint probability density $f(\mathbf{y})$ of the components of $\mathbf{y}$ factorize as

$$f(\mathbf{y}) = \prod_i f(y_i). \quad (5)$$

The functions $f(y_i)$, $i = 1, \dots, N$ are the estimates of the probability densities of the individual s_i -components. This is sometimes referred to as a factorial code [21].

The matrix $\mathbf{W}$ is determined using a statistical approach. Training data is presented to an unsupervised algorithm optimizing statistical independence of the components of $\mathbf{s}$ by iteratively adjusting an initially randomly chosen matrix $\mathbf{W}$. The training data are considered realizations of random vector $\mathbf{X}$. There exist several algorithms performing ICA [58–63]. We have used Hyvärinens FastICA algorithm [62,63], which is briefly discussed in appendix A.

The ICA basis functions are data dependent in the sense that they are learned from the training data at hand, and they will be different for different training data. The basis functions can be considered as image building blocks, capturing the inherent features of the training data. Several authors have applied ICA in order to reveal the features of natural images [21,25,24,26], that is, images void of any man made structures. In that case, the training data must be generated from images considered to belong to the class of natural images. When modeling so-called man-made images, images of buildings, cars, etc. are used to generate training data, as in [26]. We are interested in ICA basis functions which can capture the inherent properties of texture, thus in this paper the training data are generated from textured images.

2.1 Independent component analysis of natural images

In recent years it has been shown that ICA applied to natural images generates a very interesting set of basis functions [21,25,24,23,26]. Barlow [64,65] studied coding principles which could predict the formation of localized, oriented receptive fields in the visual cortex [66]. He suggested that the cortical feature detectors might be the end result of a redundancy reduction process, in which the activation of each feature detector should be as statistically independent as possible from the activation of the others. This implies a factorial code. Bell and Sejnowski [21] reconsidered

the theories of Barlow and developed an information maximization network, which performs ICA, and achieved results as predicted by Barlow’s theory. They found ICA basis functions which were localized in space, e.g. a single edge at a certain location, and of a multiscale nature [21]. Their findings were quite close to those obtained by the related sparseness maximization network of Olshausen and Field [67].

[Fig. 2 about here.]

Similar findings have also been reported in e.g. [23,26,25]. The ICA basis functions have been shown to be localized Gabor-like functions of various orientations, spatial frequencies and phases, and to bear a striking resemblance to the receptive fields of neurons in the primary visual cortex [25]. Figure 2 shows 6 examples of ICA basis functions created using the same natural images as were used in [23,26]. The characteristics mentioned above, can be clearly noted.

In [26] the ICA representation of man-made scenes was studied. These basis functions were found to have continuous lines and edges, and appear quite different from those obtained from natural images.

3 The ICA representation of texture

In this section we discuss in detail how ICA can be used to obtain a data dependent filter bank representation of textured images. For comparison we also briefly review the Gabor filter bank representation, and give some comparative remarks.

3.1 Learning methodology

In order to construct ICA basis functions of size $(m \times m)$, the columns $\mathbf{a}_i$, $i = 1, \dots, N$, of $\mathbf{A}$ in equation (3) must be of size $(M \times 1)$, where $M = m^2$. This will also be the case for the training data $\mathbf{x}$. In our case, we have available a set of 20 textured images, and each $\mathbf{x}$ is generated by extracting an $(m \times m)$ image patch from one of the training textures, and representing it as a column vector. We generate a large number of training vectors (in our case 15000), ensuring that each training texture provides an equal amount of training vectors.

Note that the representation of $(m \times m)$ image patches as column vectors destroys part of the two-dimensional correlations that help characterize texture. We try to compensate for this by repeatedly rotating the textured training images when extracting the image patches. Therefore, before the selection of training patches, a texture image is rotated a random angle θ , where $\theta \in [0^\circ, 180^\circ]$. Each texture image is rotated several angles to give a number of patches from different orientations, and every sample vector is taken from a random location in the rotated image. Also, to

ensure that no texture dominates the basis set, each training texture image is normalized and made zero mean prior to the sampling. This results in a zero ensemble mean of the training vectors.

In addition, every training vector $\mathbf{x}$ is subtracted its individual local mean value. If this is not done, one of the resulting basis functions will represent the mean intensity value of the training data, which is of little value in the filter bank context. Subtraction of the local mean value has the effect that one of the PCA components gets the variance zero [22]. Therefore, without losing any information, we may reduce the dimension in PCA space by one, and hence, the maximum number of basis functions we need to estimate is $N = M - 1$.

In summary, the generation of training data includes the following computations:

- Normalize training texture images and make them zero mean
- Sampling loop: *for $i = 1$ to number-of-images*
 - * Rotation loop: *for $j = 1$ to number-of-directions*
 - Rotate training texture number i a random angle θ_j
 - Select the given number of $(m \times m)$ sized patches from random locations
 - Subtract from each patch its mean value
 - Represent samples as columns, and store in a matrix consisting of training vectors

The computations described above result in an ensemble of training vectors which are presented to the FastICA algorithm, after additional preprocessing by PCA. After convergence of the algorithm, matrix $\mathbf{W}$ has been learned from the available training data. Matrix $\mathbf{A}$ is obtained by finding the inverse (or pseudo inverse) of $\mathbf{W}$. As a last step, the columns $\mathbf{a}_i$, $i = 1, \dots, N$, are re-shaped into size $(m \times m)$ to constitute the basis functions. These are now the impulse responses of our filter bank.

It is clear that it is computationally demanding to learn ICA basis functions even of a relatively modest size. For example, $m = 12$ requires the ICA algorithm to handle training data of size (144×1) , leading to a maximum of 143 basis functions. For $m = 256$ the data vector is of size (65536×1) leading to maximum 65535 basis functions. The latter is clearly computationally prohibitive. Therefore, in our segmentation experiments we limit the size of the basis functions to (12×12) , and we also apply extensive dimension reduction as provided by PCA.

3.2 Characteristics of ICA filter bank

[Fig. 3 about here.]

[Fig. 4 about here.]

All the textures we use for training the ICA basis functions are taken from the Brodatz album [68]. Figure 3 (a) shows the 20 textures, they are all of size (640×640) . Figure 3 (b) shows an example of (12×12) ICA basis functions learned by the method outlined above. The dimension are reduced by PCA, resulting in a total of 50 functions. The basis functions are shown in decreasing order of their dominant frequency component.

We observe that a majority of the functions are localized in spatial frequency and orientation. That is, they seem to exhibit a kind of sinusoidal wave structure of a specific spatial frequency and to be oriented in a specific direction. Figure 4 (a) shows a scatter plot of the dominating frequency component versus the orientation (in degrees) for the filters of figure 3. The dominating frequency components range from approximately 0.05 Hz to about 0.38 Hz . The orientations of the functions vary significantly, covering a number of angles in the range 0° to 180° . We note that the basis functions are fairly uniformly spread in frequency and orientation.

We also observe that some of the basis functions appear to have a checkerboard-like intensity variation. See e.g. functions number 13, 16, 17, 18 or 19 of figure 3, where function number 1 is to the top left of figure 3 and function number 50 to the bottom right, read row-wise. This particular attribute obviously occurs because several of the training textures exhibit a checkerboard-like structure.

Figure 4 shows examples of the frequency domain representation of ICA basis functions number 9 (b) and number 13 (c) of figure 3. We observe that function number 9 has a single frequency component, oriented approximately 45° from the horizontal direction. The single dominating frequency component is about 0.2 Hz . The frequency contents of function number 13 is significantly different. In this case the ICA basis function seems to have several frequency components, oriented in different directions. This gives rise to the kind of checkerboard-like structure we observe in the spatial domain.

We note that the region of support of the impulse responses of the ICA basis functions in figure 3 are not localized to a certain region in space, but they extend throughout the whole patch size. Since textures typically consist of repeating patterns covering the whole image, this can be expected when textures are used as training data. This is in contradiction to ICA basis functions obtained from natural images, which typically have a region of support which is localized in space, like e.g. an edge.

[Fig. 5 about here.]

Next, we address the question of how well the information contents of an image is preserved when the number of ICA basis functions is reduced. Consider retaining only l basis functions of a (8×8) sized learning set, where l is less than or equal to 63. Denote the mixing matrix $\mathbf{A}_l$ and the de-mixing matrix $\mathbf{W}_l$. We project a data vector $\mathbf{x}$ into the l -dimensional ICA space by the transformation

$$\mathbf{y}_l = \mathbf{W}_l \mathbf{x}, \tag{6}$$

and reconstruct an estimate $\hat{\mathbf{x}}_l$ of $\mathbf{x}$ as

$$\hat{\mathbf{x}}_l = \mathbf{A}_l \mathbf{y}_l. \quad (7)$$

How close is $\hat{\mathbf{x}}_l$ to $\mathbf{x}$? Figure 5 (b), (c) and (d) shows the result of estimating $\mathbf{x}$ using $l = 10$, $l = 30$ and $l = 63$. The original $\mathbf{x}$ is also shown in figure 5 (a). We observe that even 10 basis functions maintains the basic structure of the image, yielding a normalized squared error (NSE) of $NSE \approx 0.4$. The reconstruction is of course better for 30 basis functions, with $NSE \approx 0.1$. Using all 63 basis functions yields nearly perfect reconstruction, resulting in an $NSE \approx 0$.

The above experiment shows that the basic characteristics of textured images are represented in an ICA basis of reduced dimensions (see Appendix for details).

3.3 The Gabor filter bank

The impulse response of an even-symmetric Gabor filter is given by

$$h(x, y) = g[q(x, y), w(x, y)] \cos[2\pi f_0 q(x, y)], \quad (8)$$

where

$$g(x, y) = \exp \left\{ -\frac{1}{2} \left[\frac{x^2}{\sigma_x^2} + \frac{y^2}{\sigma_y^2} \right] \right\}, \quad (9)$$

$$q(x, y) = \cos(\Phi_0)x + \sin(\Phi_0)y, \quad (10)$$

$$w(x, y) = -\sin(\Phi_0)x + \cos(\Phi_0)y. \quad (11)$$

We note that $g(x, y)$ is the Gaussian envelope with space constants σ_x and σ_y along the x and y axes, respectively. The Gaussian envelope is rotated an angle Φ_0 with respect to the x -axis (the 0° orientation), and multiplied by a sinusoidal plane wave of spatial frequency f_0 , yielding $h(x, y)$.

The frequency domain representation of equation (8) is given by

$$\begin{aligned} \frac{H(u, v)}{2\pi\sigma_x\sigma_y} = & \exp \left[-\frac{(q(u, v) - f_0)^2}{2\sigma_u^2} - \frac{w^2(u, v)}{2\sigma_v^2} \right] \\ & + \exp \left[-\frac{(q(u, v) + f_0)^2}{2\sigma_u^2} - \frac{w^2(u, v)}{2\sigma_v^2} \right], \end{aligned} \quad (12)$$

where $\sigma_u = 1/(2\pi\sigma_x)$ and $\sigma_v = 1/(2\pi\sigma_y)$. Jain and Farrokhnia [49] performed filtering by multiplying $H(u, v)$ of equation (12) with the 2-D Fourier transform of the composite textured image.

For the Gabor filter defined by equation (12), the half-peak magnitude frequency bandwidth, B_r , and orientation bandwidth, B_θ , are given by [49];

$$B_r = \log_2 \left(\frac{f_0 + \sqrt{2 \ln 2} \sigma_u}{f_0 - \sqrt{2 \ln 2} \sigma_u} \right), \quad (13)$$

$$B_\theta = 2 \tan^{-1} \left(\frac{\sqrt{2 \ln 2} \sigma_v}{f_0} \right), \quad (14)$$

where B_r is in octaves and B_θ is in degrees. B_r is often chosen to be one octave, while the orientation bandwidth is chosen to be 45° . Thus, for any given f_0 , σ_u and σ_v are given by

$$\sigma_u = \frac{f_0}{3\sqrt{2 \ln 2}}, \quad (15)$$

$$\sigma_v = \frac{\tan 22.5^\circ f_0}{\sqrt{2 \ln 2}}, \quad (16)$$

respectively.

[Fig. 6 about here.]

The impulse response and frequency content of a typical Gabor filter is shown in figure 6. Note that the region of support of the impulse response is precisely localized in space, and the frequency response is correspondingly localized in frequency. The size of the region of support of the impulse response is determined by the width of the Gaussian envelope.

For an image of width N_c pixels, the following normalized discrete radial center frequencies, f_0 , have been suggested for a bank of Gabor filters [40],

$$\frac{1\sqrt{2}}{N_c}, \frac{2\sqrt{2}}{N_c}, \frac{4\sqrt{2}}{N_c}, \dots, \frac{\frac{1}{4}N_c\sqrt{2}}{N_c}. \quad (17)$$

The above frequency choice ensures that the pass-band of the filter with the highest frequency falls below the Nyquist frequency. For some textures the lowest radial frequencies of equation (17) are not very useful because they capture features too coarse to represent textural variations in an image [57], and are therefore often omitted. Hence, for a (256×256) image the following radial frequencies are used;

$$\frac{\sqrt{2}}{2^6}, \frac{\sqrt{2}}{2^5}, \frac{\sqrt{2}}{2^4}, \frac{\sqrt{2}}{2^3}, \frac{\sqrt{2}}{2^2}. \quad (18)$$

The filters are oriented along directions 0° , 45° , 90° and 135° .

3.4 Comparison between ICA and Gabor filters

In this subsection we point out some apparent differences between the traditional Gabor filters and our ICA filters.

The Gabor transform is a special case of the window Fourier transform (short-time Fourier transform) [37], where the window function is Gaussian. The window Fourier transform is a local analysis of the frequency content of an image. The analysis is limited by the effective width of the filter in the spatial domain or equivalently by its bandwidth in the frequency domain. Hence, from equations (15) and (16) the size of the region of support of a Gabor filter is effectively inversely related to its center frequency. Gabor filters have the important property of having optimal joint localization, or resolution, in both the spatial and the spatial frequency domain [49].

In contrast, the ICA filters are finite support filters whose size is determined by the size of the image patches selected. They are data dependent in that they have been constructed from the available training data. This allows the ICA filters to vary in shape and frequency contents as can be noted from figure 3. The Gabor filters are characterized by their orientation and their center frequency. The ICA filters shown in figure 3 are richer, they may be frequency - and orientation selective like Gabor filters (see e.g. filter 9), but they may also realize multiple pass-bands of multiple orientations (see e.g. filter 13).

4 Generation of features for texture segmentation

To be able to automatically segment a multi-textured image into its distinct segments, it is necessary to generate representatives of each pixel which can capture the inherent nature of the local pattern surrounding it. These representatives usually form vector valued functions of the image pixels, denoted as feature vectors.

If the textures have sufficiently different structure and the feature vectors are well designed, they will form compact clusters in feature space, each cluster corresponding to a distinct texture region in the multi-textured input image. The goal is to reveal the clusters in feature space by the use of some clustering algorithm and subsequently assign pixels to their appropriate texture class.

The basic idea in filter based texture segmentation has already been briefly described previously. We filter the original image through the bank of filters, each with specific frequency - and orientation characteristics. If the bank consists of N filters, the result is N filtered images of the same size as the input image.

[Fig. 7 about here.]

Each filter responds to specific texture properties. Thus, each filtered image will have high energy in regions corresponding to textures which are tuned to the filter,

and low energy corresponding to textures which are not tuned to that filter. Then, the approach taken is to apply a non-linearity, such as squaring, followed by a smoothing operation to the filtered images. The resulting images are called feature images, and the feature vectors are formed by collecting corresponding pixel values from each feature image in a vector. The experimental setup described here is shown in figure 7.

4.1 Creating feature images

When creating features based on an energy measure, the objective of the non-linearity and smoothing operations, is to estimate the energy in a local region of the filter output. The most commonly used non-linearities are the magnitude function $|\cdot|$ [41,43,48,69], the squaring function $(\cdot)^2$ [56,53,51], or the rectified sigmoid function $|\tanh(\alpha\cdot)|$ [49]. We use the squaring function.

It is furthermore important to determine a proper size and shape of the smoothing window. If the smoothing window is too small, the estimate of the local energy will be unreliable, while if the window is too large, we will smooth out boundaries between texture regions. A Gaussian window is likely to result in more accurate localization of texture boundaries rather than a rectangular window [49]. Jain and Farrokhnia [49] suggested to let the variance of the Gaussian smoothing window be dependent on the center frequency of the Gabor filter according to $\sigma = \frac{1}{2\sqrt{2}f_0}$. We find that using $\sigma = \frac{2}{\sqrt{3}f_0}$ yields better results in our experiments for both the ICA and the Gabor filter banks. Also, the size of the window can be fixed to a preselected value, or determined automatically. We choose the size of the window to be $(l \times l)$, where $l = 2\sigma$. Thus, the Gaussian window width is also different for each filter in the filter bank.

In addition, as a last step, we apply a logarithmic normalizing non-linearity. The combination of squaring and logarithmic normalization was found by Unser and Eden [70] to give very good texture segmentation performance. We use this normalization. Denoting the k 'th filtered image by $r_k(x, y)$, the k 'th feature image $e_k(x, y)$ is then given by

$$e_k(x, y) = \log \left(\frac{1}{l^2} \sum_{a,b \in W} r_k^2(x-a, y-b) W(a, b) \right), \quad (19)$$

where W is the Gaussian window centered at pixel coordinates (x, y) .

4.2 Filter selection

Filter selection refers to the problem of selecting a subset of the feature generating filters which exhibit high information packing in the sense of having high texture

separability performance. The problem is; which filters to choose, and how many.

Referring to figure 3, we observe that some of the ICA basis functions look quite similar. These functions will have a high degree of similarity also in their frequency domain representations, and are expected to yield almost identical outputs when used to filter an input textured image. This means that the features based on these filters will provide us with essentially the same information and thus bring redundancy into the feature vectors.

Our first filter selection step is to find the subset of J filters with the smallest degree of overlap in the frequency domain. We simply form the frequency representations, i.e. the transfer functions, F_i and F_j of two filters i and j , and measure their frequency overlap by calculating $|F_i - F_j|$. This is done pairwise for all the filters, and the J filters having mutually the smallest overlap are chosen. All filters are normalized in energy before the selection procedure.

Our next step is to examine the feature images of all the J remaining filters, to obtain a ranking based on whether the features possess variability over the textured image or not. If a feature image does not have variability, the feature does not provide local information. In doing this ranking, we use the F -test proposed in [71]. We divide each feature image into R equally sized image regions, Ω_r , $r = 1, \dots, R$, and compute the residual sum of squares (RSS) in each of the regions. The RSS of feature image k in region Ω_r , is defined by

$$RSS_{kr} = \sum_{i,j \in \Omega_r} [e_k(i,j) - m_{kr}]^2, \quad (20)$$

where $e_k(i,j)$ is the value of feature no. k in pixel (i,j) , and m_{kr} is the mean of the feature in the region Ω_r . We form the total RSS_{kT} as the sum

$$RSS_{kT} = \sum_{r=1}^R RSS_{kr}. \quad (21)$$

This value is compared to the RSS_k computed over the whole feature image $e_k(x,y)$,

$$RSS_k = \sum_{i,j} [e_k(i,j) - m_k]^2, \quad (22)$$

where m_k is the mean value of the k -th feature image. If RSS_{kT} is close to RSS_k , the feature does not provide local information.

The F -test is based on [71]

$$F_k = \frac{(N - R) (RSS_k - RSS_{kT})}{(R - 1) RSS_{kT}}, \quad (23)$$

where N is the total number of pixels in each of the regions Ω_r , $r = 1, \dots, R$.

When it comes to the number of filters to be used in the filter bank, one possibility is to use those filters yielding values of F_k above a significance-threshold U_T . The threshold is determined from the feature corresponding to the largest F value.

4.3 Clustering in feature space

We use the well-known K -means algorithm [29,37] to cluster the feature vectors. This is a simple algorithm, which also has been widely applied in the texture segmentation context. The implementation, i.e. the initialization of the cluster prototypes, may be done in several ways, and the clustering result is often dependent on which initialization procedure is used. We initialize the prototypes by randomly drawing prototypes from the set of feature vectors. This method has the disadvantage that it may converge to a local minimum in the K -means cost function, which in our case is the sum of squared prototype distance. To avoid this problem, we run the algorithm several times, and choose the best clustering as our final result.

A well known problem with the K -means algorithm is that it requires a priori knowledge of the number of clusters to be formed. For a fully automatic texture segmentation algorithm it is therefore of crucial importance to be able to obtain reliable estimates of this number before running the K -means algorithm. There exists a variety of different cluster validity indices [72,73] for this purpose. In this paper, the main objective is to introduce ICA generated features, and analyze texture segmentation based on these. We therefore assume that we have a priori knowledge of the number of clusters.

5 Segmentation experiments

In this section, we report results on texture segmentation using the ICA filters. The filters are all learned from the training textures in Figure 3 (a) using (12×12) sized patches. We have previously pointed out the important property of the ICA filters being data dependent. This means that they are sensitive to the training data, and to emphasize this fact, we have tested two basis sets. The first set, shown in Figure 8 (a), is the 20 basis functions with the least frequency overlap, selected among the 50 filters previously shown in figure 3 (b). They are ordered according to descending F -value (see section 4.2). Note that both filters number 9 and 13 of figure 3, with frequency responses shown in figure 4 (a) and (b), are among the selected filters (filters number 17 and 13 in figure 8 (a), respectively). The second set, shown in Figure 8 (b), is generated from the same training textures, but for this set we did not normalize the textured image energy prior to the ICA basis generation.

The segmentation performance of these two sets are compared to that of a Gabor filter bank consisting 20 filters. The Gabor filters have radial frequencies according to equation (18), and are oriented along the directions $0^\circ, 45^\circ, 90^\circ$ and 135° . In

all cases, the width of the Gaussian smoothing windows is given by $\sigma = \frac{2}{\sqrt{3}f_0}$, where f_0 in the ICA cases is an estimated dominating frequency. The segmentation is performed using the K -means algorithm to cluster the energy feature vectors obtained by filtering the input texture through the respective filter banks.

[Fig. 8 about here.]

[Fig. 9 about here.]

Two multi-textured input images (256×256) are used. Figure 9 (b) shows the first, which is composed of two different texture patterns. The performance of an ICA filter bank is obviously depending on how many filters are used. This dependency has been depicted in Figure 9 (a), where we have plotted the percentage of mis-classified pixels as function of the number of filters. The dashed curve corresponds to the filters of normalized input textures, and the solid corresponds those of the non-normalized. When extending the filter bank, we include more and more filters following the order they are presented in Figures 8 (a) and (b). We note that the first set almost always performs the best, and an optimum result is obtained with 7 filters, resulting in an error percentage of 2.9%. The second set has its best performance using 20 filters, achieving an error percentage of 3.5%. The segmented images are displayed in Figures 9 (c), (d) and (e), where the white solid curve marks the true boundary between the two texture patterns. We note that both the ICA sets in this case outperform the Gabor filter bank, which achieves a best error rate of 13.8%.

The second multi-textured input image consists of five texture patterns, and is displayed in Figure 10 (b). Figure 10 (a) show how the mis-classification rate changes with the number of filters in the ICA filter banks. In this case the best results are obtained using 13 filters for the first set (normalized input textures), and 11 filters for the second set. The error rates are 2.8% and 3.0%, respectively. Also in this case, the ICA filter banks perform better than the Gabor filter bank, although the difference is not as large as in the previous experiment. The best result for the Gabor filterbank is in this case 3.6%. The segmented images are depicted in Figures 10 (c), (d) and (e).

[Fig. 10 about here.]

6 Conclusions

We have presented independent component analysis (ICA) of textured images as a computational technique for creating a new data dependent filter bank for use in texture segmentation.

The ICA filters can be used as basis functions in an image model, in which an image is expressed as a weighted sum of the impulse responses. These basis functions are obtained in an unsupervised learning process, where the criterion for learning is that the weighting components should be as statistically independent as possible.

The training data are in this application gathered from a set of textured images. Thus, the ICA filters are data dependent.

We have demonstrated that the ICA filters are able to capture the inherent properties of textured images. A majority of the functions are localized in spatial frequency and orientation. That is, they seem to exhibit a kind of sinusoidal wave structure of a specific spatial frequency and direction. Such filters can be interpreted as oriented band-pass filters. Interestingly, some of the ICA filters have frequency responses with several components, orientated in different directions. The new ICA filter bank is similar to the Gabor filter bank, but it seems to be richer in the sense that some filters have more complex frequency responses.

The above properties allow us to use the ICA filter bank to create energy features for effective texture segmentation. Our experiments using multi-textured images show that the ICA filter bank yields similar or better segmentation results than the Gabor filter bank.

A ICA theory

In this appendix we give a short review of basic ICA theory. For a comprehensive treatment we refer to e.g. [4].

A-1 The linear ICA model

We consider the statistical model in which the observed signal $\mathbf{x}$ is given as a linear mixture of some sources $\mathbf{s}$

$$\mathbf{x} = \mathbf{A}\mathbf{s}. \tag{i}$$

The goal is to estimate the de-mixing $\mathbf{W}$, such that the vector $\mathbf{x}$ is transformed into

$$\mathbf{y} = \mathbf{W}\mathbf{x}, \tag{ii}$$

yielding as mutually statistically independent components of $\mathbf{y}$ as possible (note that we have suppressed the time dependency). The estimation of $\mathbf{W}$ is done by presenting training vectors to an unsupervised algorithm optimizing independence of the components in $\mathbf{y}$. The vector $\mathbf{y}$ is thus an estimate of the true sources $\mathbf{s}$. The training vectors $\mathbf{x}$ are represented as samples of the random vector $\mathbf{X}$.

A-2 Preprocessing

Two preprocessing steps are common in ICA [21,25].

First, the data is centered,

$$\mathbf{x} := \mathbf{x} - E\{\mathbf{x}\}. \quad (\text{iii})$$

Thus, in ICA, we deal with zero mean data. Note that this means that $E\{\mathbf{s}\} = 0$.

The next step is to remove the second order statistical dependence in the data by whitening. Whitened data are uncorrelated and have unit variance. This is achieved by using a whitening matrix $\mathbf{V}$ such that

$$\mathbf{z} = \mathbf{V}\mathbf{x}, \text{ and } E\{\mathbf{z}\mathbf{z}^T\} = \mathbf{I}. \quad (\text{iv})$$

In our case we apply the common principal component analysis (PCA) whitening, i.e. $\mathbf{V}_{\text{PCA}} = \mathbf{D}^{-\frac{1}{2}}\mathbf{E}^T$, using the eigenvalue decomposition $\mathbf{E}\mathbf{D}\mathbf{E}^T = E\{\mathbf{x}\mathbf{x}^T\}$ of the covariance matrix of $\mathbf{x}$. $\mathbf{E}$ is here the matrix containing the eigenvectors of $E\{\mathbf{x}\mathbf{x}^T\}$, and $\mathbf{D}$ holds the corresponding eigenvalues on its diagonal.

Having centered and whitened the data $\mathbf{x}$, the task is to find matrix $\mathbf{B}$, such that

$$\mathbf{y} = \mathbf{B}\mathbf{z} = \mathbf{B}\mathbf{D}^{-\frac{1}{2}}\mathbf{E}^T\mathbf{x}. \quad (\text{v})$$

Note that $\mathbf{W} = \mathbf{B}\mathbf{D}^{-\frac{1}{2}}\mathbf{E}^T$. If the training data $\mathbf{x}$ is of dimension $(M \times 1)$, then in this basic ICA model, matrices $\mathbf{B}$, $\mathbf{D}$ and $\mathbf{E}$ are all of dimension $(M \times M)$, and the dimension of $\mathbf{y}$ is also $(M \times 1)$.

A-3 Independence optimization

There exist several algorithms performing ICA [58–63]. We have used Hyvärinens FastICA algorithm [62,63]. The FastICA algorithm constrains $\mathbf{B}$ to be orthogonal (orthonormal). It outputs the independent components one at a time by estimating matrix $\mathbf{B}$ row by row. That is, it first finds component $y_1 = \mathbf{b}_1^T \mathbf{z}$, then $y_2 = \mathbf{b}_2^T \mathbf{z}$ and so on. Here $\mathbf{b}_i^T$, $i = 1, \dots, M$, is the i 'th row of $\mathbf{B}$. Each new $\mathbf{b}_i^T$ to be estimated is constrained to be orthogonal to all the previously estimated $\mathbf{b}_j^T$, $j < i \leq M$.

Each vector $\mathbf{b}_i^T$ is optimized in an iterative manner. Consider estimating the i 'th independent component, s_i . Denote the estimate of s_i at a certain iteration step by

y_i . Thus $y_i = \hat{\mathbf{b}}_i^T \mathbf{z}$, and by equations (iv) and (i),

$$y_i = \hat{\mathbf{l}}_i^T \mathbf{s} = \sum_{j=1}^M \hat{l}_{ij} s_j, \quad (\text{vi})$$

where $\hat{\mathbf{l}}_i^T = \hat{\mathbf{b}}_i^T \mathbf{V}_{\text{PCA}} \mathbf{A}$. Since the individual s_j , $j = 1, \dots, M$, have unit variances by equations (iv) and (v), then by the central limit theorem, it is clear that y_i will be more Gaussian than any of the individual s_j , since a sum of independent variables with finite variance will produce a more Gaussian variable. The strategy employed is to force $y_i = \hat{\mathbf{b}}_i^T \mathbf{z}$ towards maximum non-Gaussianity because this would in fact imply adjusting $\hat{\mathbf{l}}_i^T$ until it has only one non-zero element (± 1), namely the element multiplied to the most non-Gaussian s_j . This enables us to estimate the i th independent component.

This discussion also brings to light why the true s_j , $j = 1, \dots, M$ (except possibly one) need to have non-Gaussian statistics. If this is not the case, it would be impossible to separate the signals, since a sum of Gaussians remains Gaussian.

The iterative procedure is accomplished by updating each row $\mathbf{b}_i^T$, $i = 1, \dots, M$, of $\mathbf{B}$ by [62,63]

$$\mathbf{b}_i^+ = E\{\mathbf{z} g_1(\mathbf{b}_i^T \mathbf{z})\} - E\{g_1'(\mathbf{b}_i^T \mathbf{z})\} \mathbf{b}_i, \quad (\text{vii})$$

followed by a normalization. In this equation $\mathbf{b}_i^+$ denotes the updated vector, and $g_1(u)$ is a non-linearity. We use $g_1(u) = \tanh(u)$ [62,63].

A-4 Reducing dimension

PCA enables us to optimally (in a squared error sense) reduce the dimension of the data, by using only the eigenvectors of the largest eigenvalues in the matrix $\mathbf{V}_{\text{PCA}}$.

When reducing dimension from $\mathbf{x} \sim (M \times 1)$ to $\mathbf{y} \sim (N \times 1)$ where $N < M$, the model becomes

$$\mathbf{y}_N = \mathbf{B} \mathbf{z} = \mathbf{B} \mathbf{D}_N^{-\frac{1}{2}} \mathbf{E}_N^T \mathbf{x}, \quad (\text{viii})$$

where $\mathbf{D}_N$ is the matrix containing the N largest eigenvalues, and $\mathbf{E}_N$ holds the corresponding eigenvectors. Note that in this case $\mathbf{B} \sim (N \times N)$, $\mathbf{D}_N \sim (N \times N)$ and $\mathbf{E}_N \sim (M \times N)$. Thus $\mathbf{y} \sim (N \times 1)$.

The reconstruction of image $\mathbf{x}$ using dimensionality reduction is $\hat{\mathbf{x}}_{\text{PCA}} = \mathbf{E}_N \mathbf{z}_N$, with $\mathbf{z}_N = \mathbf{E}_N^T \mathbf{x}$, and the mean square error between the image $\mathbf{x}$ and it's PCA

reconstruction given by

$$E\{\|\mathbf{x} - \hat{\mathbf{x}}_{\text{PCA}}\|^2\} = E\{\|\mathbf{x} - \mathbf{E}_N \mathbf{E}_N^T \mathbf{x}\|^2\} = \sum_{i=N+1}^M \lambda_i. \quad (\text{ix})$$

λ_i , $i = 1, \dots, M$, are the eigenvalues sorted in decreasing order, and M is the original dimension of the data.

It is easy to show that the mean square error when using dimensionality reduction in the ICA transform is the same as in equation (ix). The ICA transform (with reduced dimension) is given by equation (viii). The reconstructed image, $\hat{\mathbf{x}}_{\text{ICA}}$, is given as

$$\hat{\mathbf{x}}_{\text{ICA}} = \mathbf{A} \mathbf{y}_N = \mathbf{E}_N \mathbf{D}_N^{\frac{1}{2}} \mathbf{B}^T \mathbf{y}_N = \mathbf{E}_N \mathbf{D}_N^{\frac{1}{2}} \mathbf{B}^T \mathbf{B} \mathbf{D}_N^{-\frac{1}{2}} \mathbf{E}_N^T \mathbf{x} = \mathbf{E}_N \mathbf{E}_N^T \mathbf{x}, \quad (\text{x})$$

We utilize these results in subsection 3.2 when calculating the normalized square error (NSE) between an image $\mathbf{x}$ and its estimate $\hat{\mathbf{x}}_{\text{ICA}}$. We normalize the squared error with the sum of all eigenvalues of the covariance matrix $E\{\mathbf{x}\mathbf{x}^T\}$ of the training data, such that the NSE is given by,

$$NSE = \frac{\|\mathbf{x} - \hat{\mathbf{x}}_{\text{ICA}}\|^2}{\sum_{i=1}^M \lambda_i} \in [0, 1]. \quad (\text{xi})$$

References

- [1] C. Jutten and J. Héroult. Blind separation of sources, part I: An adaptive algorithm based on neuromimetic architecture. *Signal Processing*, 24:1–10, 1991.
- [2] P. Comon. Independent component analysis - a new concept? *Signal Processing*, 36:287–314, 1994.
- [3] C. Jutten. Source separation: from dusk till dawn. In *Proc. 2nd Int. Workshop on Independent Component Analysis and Blind Source Separation (ICA2000)*, pages 15–26, Helsinki, Finland, 2000.
- [4] A. Hyvärinen, J. Karhunen, and E. Oja. *Independent Component Analysis*. Wiley Series on Adaptive and Learning Systems for Signal Processing, Communications, and Control, Simon Haykin Series Editor. John Wiley and Sons, Inc., 2001.
- [5] P. Comon, C. Jutten, and J. Héroult. Blind separation of sources, part II: Problems statement. *Signal Processing*, 24:11–20, 1996.
- [6] E. Sorouchyari. Blind separation of sources, part III: Stability analysis. *Signal Processing*, 24:21–29, 1991.

- [7] T.P. Jung, C. Humphries, T.-W. Lee, S. Makeig, M.-J. McKeown, V. Iragui, and T. Sejnowski. Extended ICA removes artifacts from electroencephalographic recordings. In *Advances in Neural Information Processing Systems*, volume 10, MIT Press, 1998.
- [8] R. Vigário. Extraction of ocular artifacts from EEG using independent component analysis. *Electroenceph. Clin. Neurophysiol.*, 103(3):395–404, 1997.
- [9] R. Vigário, V. Jousmäki, M. Hämäläinen, R. Hari, and E. Oja. Independent component analysis for identification of artifacts in magnetoencephalographic recordings. In *Advances in Neural Information Processing Systems*, volume 10, pages 229–235, MIT Press, 1998.
- [10] R. Vigário, J. Särelä, V. Jousmäki, M. Hämäläinen, and E. Oja. Independent component approach to the analysis of EEG and MEG recordings. *IEEE Trans. Biomedical Engineering*, 47(5):589–593, 2000.
- [11] R. Cristescu, J. Joutsensalo, J. Karhunen, and E. Oja. A complexity minimization approach for estimating fading channels in CDMA communications. In *Proc. Int. Workshop on Independent Component Analysis and Blind Signal Separation (ICA2000)*, pages 527–532, Helsinki, Finland, June 2000.
- [12] R. Cristescu, T. Ristaniemi, J. Joutsensalo, and J. Karhunen. Blind separation of convolved mixtures for CDMA systems. In *Proc. Tenth European Signal Processing Conference (EUSIPCO2000)*, pages 619–622, Tampere, Finland, 2000.
- [13] K. Pawelzik, K.-R. Müller, and J. Kohlmorgen. Prediction of mixtures. In *Proc. Int. Conf. on Artificial Neural Networks (ICANN'96)*, pages 127–132, Springer, 1996.
- [14] T. Eltoft and Ø. Kristensen. ICA and nonlinear times series prediction for recovering missing data segments in multivariate signals. In *Proc. 3rd Int. Workshop on Independent Component Analysis and Blind Signal Separation (ICA2001)*, pages 716–721, San Diego, USA, December 2001.
- [15] S. Malaroiu, K. Kiviluoto, and E. Oja. Time series prediction with independent component analysis. In *Proc. Int. Conf. on Advanced Investment Technology*, Gold Coast, Australia, 2000.
- [16] K. Torkkola. Blind separation for audio signals - are we there yet? In *Proc. 1st Int. Workshop on Independent Component Analysis and Source Separation (ICA'99)*, pages 239–244, Aussois, France, 1999.
- [17] F. M. Ham and N. A. Faour. Infrasound signal separation using independent component analysis. In *Proc. 21st Seismic Research Symposium: Technologies for Monitoring the Comprehensive Nuclear-Test-Ban Treaty*, Las Vegas, Nevada, USA, 1999.
- [18] J.-F. Cardoso, C. Jutten, and P. Loubaton, editors. *Proc. of the 1st Int. Workshop on Independent Component Analysis and Signal Separation*, Aussois, France, January 1999.

- [19] P. Pajunen and J. Karhunen, editors. *Proc. of the 2nd Int. Workshop on Independent Component Analysis and Blind Signal Separation*, Helsinki, Finland, June 2000.
- [20] T.-W. Lee, T. P. Jung, S. Makeig, and T. Sejnowski, editors. *Proc. of the 3rd Int. Conference on Independent Component Analysis and Blind Signal Separation*, San Diego, California, USA, December 2001.
- [21] A. J. Bell and T. J. Sejnowski. The "independent components" of natural scenes are edge filters. *Vision Research*, 37:3327–3338, 1997.
- [22] J. Hurri. Independent component analysis of image data. Master's thesis, Helsinki University of Technology, 1997.
- [23] J. Hurri, A. Hyvärinen, and E. Oja. Wavelets and natural image statistics. In *Proc. Scandinavian Conf. on Image Analysis*, Lappeenranta, Finland, 1997.
- [24] J. H. van Hateren and A. van der Schaaf. Independent component filters of natural images compared with simple cells in primary visual cortex. *Proc. Royal Society ser. B*, 256:359–366, 1998.
- [25] P. Hoyer and A. Hyvärinen. Independent component analysis applied to feature extraction from colour and stereo images. *Network: Computation in Neural Systems*, 11(3):191–210, 2000.
- [26] A. Hyvärinen. Sparse code shrinkage: Denoising of nongaussian data by maximum likelihood estimation. *Neural Computation*, 11(7):1739–1768, 1999.
- [27] R. Jenssen, T. A. Øigård, T. Eltoft, and A. Hanssen. Sparse code shrinkage based on the normal inverse Gaussian density model. In *Proc. 3rd Int. Workshop on Independent Component Analysis and Blind Signal Separation (ICA2001)*, pages 212–217, San Diego, USA, December 2001.
- [28] R. M. Haralick, K. Shanmugam, and I. Dinstein. Textural features for image classification. *IEEE Transactions on Systems Man and Cybernetics*, 3(6):610–621, 1973.
- [29] S. Theodoridis and K. Koutroumbas. *Pattern Recognition*. Academic Press, 1999.
- [30] L. Davis and S. Johns and J. K. Aggrawal. Texture analysis using generalized co-occurrence matrices. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1(3):251–259, 1979.
- [31] H. Tamura, S. Mori, and T. Yamawaki. Textural features corresponding to visual perception. *IEEE Transactions on Systems, Man, and Cybernetics*, 8(6):460–473, 1978.
- [32] J. Zhang and T. Tan. Brief review of invariant texture analysis methods. *Pattern Recognition*, 35:735–747, 2002.
- [33] F. Cohen. Classification of rotated and scaled texture images using gaussian markov random field models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(2):198–202, 1992.

- [34] K. Sivakumar. Morphologically constrained GRFs: applications to texture synthesis and analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(2):148–153, 1999.
- [35] L. Fang and R. W. Picard. Periodicity, directionality, and randomness: Wold features for image modeling and retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(7):722–733, 1996.
- [36] L. Fang and R. W. Picard. Periodicity, directionality, and randomness: Wold features for perceptual pattern recognition. In *Proceedings of the International Conference on Pattern Recognition*, volume 11, pages 184–185, Jerusalem, October 1994.
- [37] M. Tuceryan and A. K. Jain. *The Handbook of Pattern Recognition and Computer Vision*, chapter 2.1, pages 207–248. World Scientific Publishing Co., by C. H. Chen, L. F. Pau and P. S. P. Wang (eds.), 2nd edition, 1998.
- [38] F. W. Campbell and J. G. Robson. Application of Fourier analysis to the visibility of gratings. *Journal of Physiology*, 197:551–556, 1968.
- [39] R. L. De Valois, D. G. Albrecht, and L. G. Thorell. Spatial-frequency selectivity of cells in macaque visual cortex. *Vision Research*, 22:545–559, 1982.
- [40] T. Randen and J. H. Husøy. Filtering for texture classification: A comparative study. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(4):291–310, 1999.
- [41] K. I. Laws. Rapid texture identification. In *SPIE Image Processing for Missile Guidance*, volume 238, pages 376–380, 1980.
- [42] K. I. Laws. *Textured Image Segmentation*. PhD thesis, University of Southern California, 1980.
- [43] J. M. Coggins and A. K. Jain. A spatial filtering approach to texture analysis. *Pattern Recognition Letters*, 3:195–203, 1985.
- [44] D. A. Pollen and S. F. Ronnen. Visual cortical neurons as localized spatial filters. *IEEE Trans. Syst. Man. Cybern.*, 13:907–916, 1983.
- [45] S. Marcelja. Mathematical description of the responses of simple cortical cells. *Journal Opt. Soc. Am.*, 70:1297–1300, 1980.
- [46] J. G. Daugman. Two-dimensional spectral analysis of cortical receptive field properties. *Vision Research*, 20:847–856, 1980.
- [47] M. R. Turner. Texture discrimination by Gabor functions. *Biol. Cybern.*, 55:71–82, 1986.
- [48] A. C. Bovik, M. Clark, and W. S. Geisler. Multichannel texture analysis using localized spatial filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12:55–73, 1990.
- [49] A. K. Jain and F. Farrokhnia. Unsupervised texture segmentation using Gabor filters. *Pattern Recognition*, 24(12):1167–1186, 1991.

- [50] O. Pichler, A. Teuner, and B. Hosticka. A comparison of texture feature extraction using adaptive Gabor filtering, pyramidal and tree structured wavelet transforms. *Pattern Recognition*, 29(5):733–742, 1996.
- [51] A. Teuner, O. Pichler, and B. J. Hosticka. Unsupervised texture segmentation of images using tunes matched Gabor filters. *IEEE Transactions on Image Processing*, 4(6):863–870, 1995.
- [52] A. Laine and J. Fan. Texture classification by wavelet packet signatures. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(11):1186–1191, 1993.
- [53] T. Chang and C.-C. J. Kuo. Texture analysis and classification with tree-structured wavelet transform. *IEEE Transactions on Image Processing*, 2(4):429–441, 1993.
- [54] N. Saito and R. R. Coifman. Local discriminant bases and their applications. *J. Mathematical Imaging and Vision*, 5(4):337–358, 1995.
- [55] A. Laine and J. Fan. Frame representations for texture segmentation. *IEEE Transactions on Image Processing*, 5(5):771–780, 1996.
- [56] M. Unser. Texture classification and segmentation using wavelet frames. *IEEE Transactions on Image Processing*, 4(11):1549–1560, 1995.
- [57] T. Randen and J. H. Husøy. Multichannel filtering for image texture segmentation. *Optical Engineering*, 33:2617–2625, 1994.
- [58] S. Amari, A. Cichocki, and H. Yang. A new learning algorithm for blind source separation. In *Advances in Neural information processing systems*, pages 757–763. MIT Press, Cambridge, MA, 1996.
- [59] A. J. Bell and T. J. Sejnowski. An information-maximization approach to blind separation and blind deconvolution. *Neural Computation*, 7:1129–1159, 1995.
- [60] J.-F. Cardoso and B. H. Laheld. Equivariant adaptive source separation. *IEEE Transactions on Signal Processing*, 44(12):3017–3030, 1996.
- [61] A. Cichocki and R. Unbehauen. Robust neural networks with on-line learning for blind identification and blind separation of sources. *IEEE Transactions on Circuits and Systems*, 43(11):894–906, 1996.
- [62] A. Hyvärinen and E. Oja. A fast fixed-point algorithm for independent component analysis. *Neural Computation*, 9:1483–1492, 1997.
- [63] A. Hyvärinen. Fast and robust fixed-point algorithms for independent component analysis. *IEEE Transactions on Neural Networks*, 10(3):626–634, 1999.
- [64] H. B. Barlow. Unsupervised learning. *Neural Computation*, 1:295–311, 1989.
- [65] J. J. Atick and A. N. Redlich. What does the retina know about natural scenes? *Neural Computation*, 4:308–320, 1992.

- [66] H. B. Barlow and D. J. Tolhurst. Why do you have edge detectors? *Optical Society of America: Technical Digest*, 23(172), 1992.
- [67] B. A. Olshausen and D. J. Field. Natural image statistics and efficient coding. *Network: Computation in Neural Systems*, 7(2), 1996.
- [68] P. Brodatz. Texture: A photographic album for artists and designers. Dover, New York 1966.
- [69] T. P. Weldon and W. E. Higgins. Designing multiple Gabor filters for multi-texture image segmentation. *Optical Engineering*, 38(9):1478–1489, 1999.
- [70] M. Unser and M. Eden. Nonlinear operators for improving texture segmentation based on features extracted by spatial filtering. *IEEE Transactions on Systems, Man, Cybernetics*, 20:804–815, 1990.
- [71] T. Kasparis, D. Charalampidis, M. Georgiopoulos, and J. Rolland. Segmentation of textured images based on fractals and image filtering. *Pattern Recognition*, 34:1963–1973, 2001.
- [72] A. K. Jain and R. C. Dubes. *Algorithms for Clustering Data*. Prentice-Hall, Englewood Cliffs, NJ, 1988.
- [73] G. W. Milligan and M. C. Cooper. An examination of procedures for determining the number of clusters in a data set. *Psychometrika*, pages 159–179, 1985.

List of Figures

1	The ICA model. An image is modeled as a weighted summation of basis images. The weights are the “independent components”.	26
2	Independent basis functions of size (32×32) learned from natural images.	27
3	Training textures and resulting ICA basis functions	28
4	Characterization of ICA filters: (a) Scatter plot of dominating frequency component versus orientation (in degrees) for the ICA basis functions of figure 3. (b), (c) Contour plots of frequency domain representation of ICA basis functions number 9 and number 13, respectively	29
5	Reconstruction of original (8×8) image using ICA basis functions. (a) Original image, (b) Using 10 basis functions yielding $NSE \approx 0.4$, (c) Using 30 basis functions, $NSE \approx 0.1$ and (d) Using 63 basis functions, $NSE \approx 0$.	30
6	Example of a Gabor filter in a (64×64) grid. Left: Spacial domain, Right: Frequency domain. Normalized frequency is $f_0 = \frac{\sqrt{2}}{2^3}$, and orientation 135° .	31
7	Experimental setup for segmenting a multi-textured input image.	32
8	The 20 most uncorrelated (12×12) ICA functions learned from (a) normalized and (b) non-normalized training textures. The basis functions are ordered from top left to bottom right according to their F -test significance.	33
9	Segmentation results. Textured image is composed of Brodatz textures $D53$ (inner pattern) and $D49$.	34
10	Segmentation results. Textured image is composed of Brodatz textures $D77$, $D55$, $D84$, $D21$ (top left to bottom right) and $D24$ (inner pattern).	35

$$\text{Image} = s_1 \text{Basis}_1 + s_2 \text{Basis}_2 + \dots + s_N \text{Basis}_N$$

Fig. 1. The ICA model. An image is modeled as a weighted summation of basis images. The weights are the “independent components”.

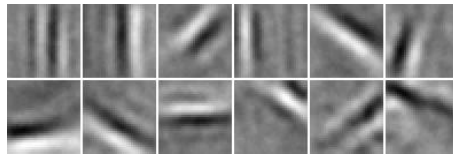
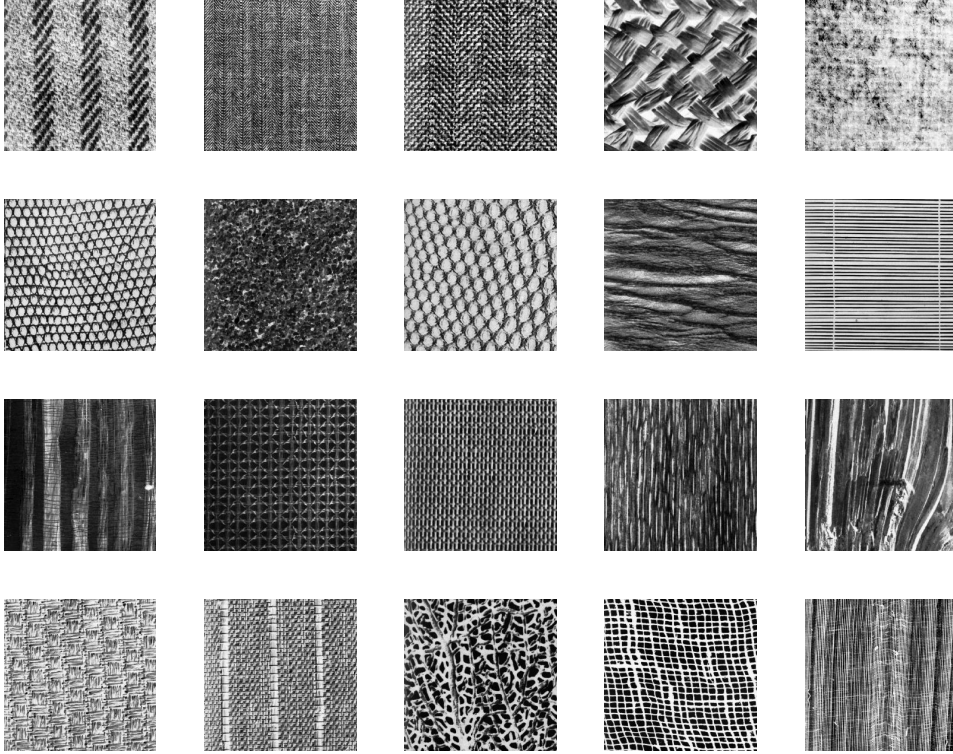
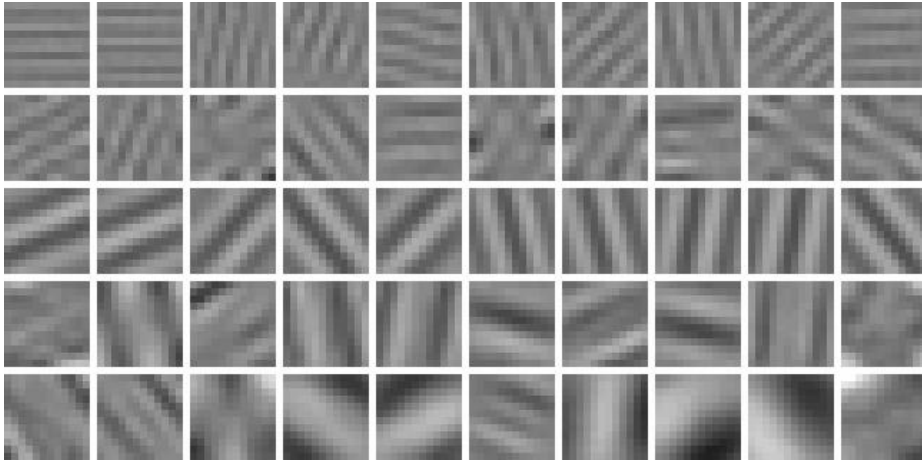


Fig. 2. Independent basis functions of size (32×32) learned from natural images.



(a) Training textures from the Brodatz album, from top left to bottom right: D_{11} , D_{16} , D_{17} , D_{18} , D_{19} , D_{22} , D_{29} , D_{35} , D_{37} , D_{49} , D_{50} , D_{52} , D_{53} , D_{68} , D_{72} , D_{82} , D_{85} , D_{87} , D_{103} and D_{105} .



(b) Example of (12×12) ICA basis functions for texture.

Fig. 3. Training textures and resulting ICA basis functions

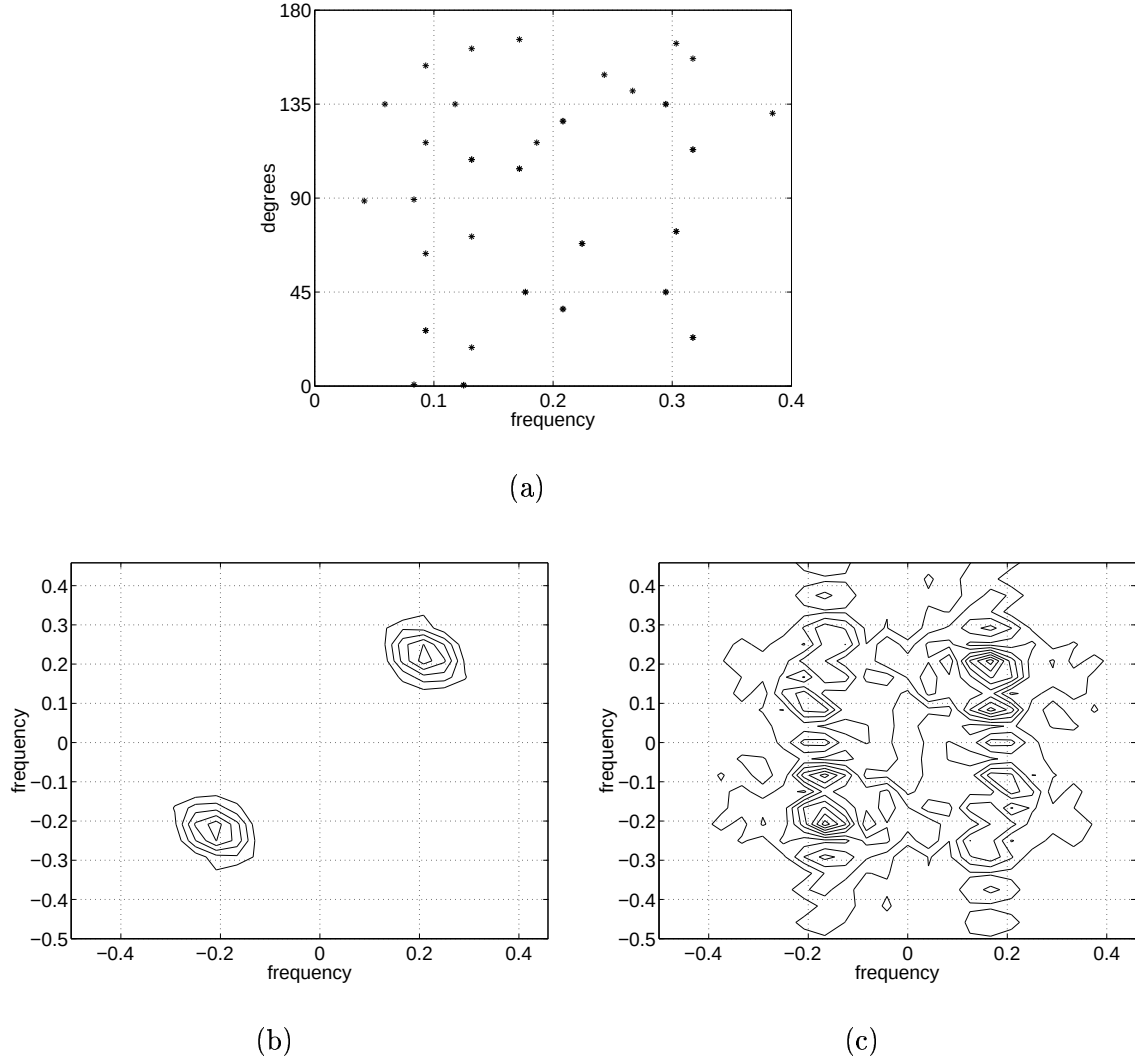
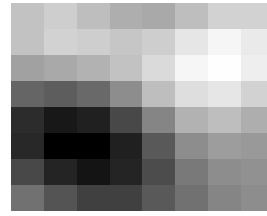


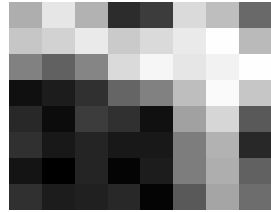
Fig. 4. Characterization of ICA filters: (a) Scatter plot of dominating frequency component versus orientation (in degrees) for the ICA basis functions of figure 3. (b), (c) Contour plots of frequency domain representation of ICA basis functions number 9 and number 13, respectively



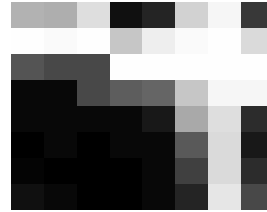
(a) Original (8×8)



(b) 10 basis functions



(c) 30 basis functions



(d) 63 basis functions

Fig. 5. Reconstruction of original (8×8) image using ICA basis functions. (a) Original image, (b) Using 10 basis functions yielding $NSE \approx 0.4$, (c) Using 30 basis functions, $NSE \approx 0.1$ and (d) Using 63 basis functions, $NSE \approx 0$.

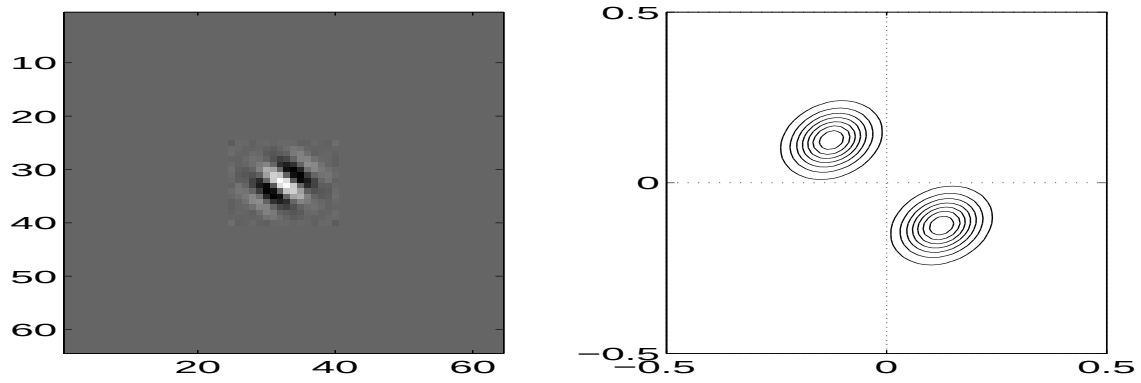


Fig. 6. Example of a Gabor filter in a (64×64) grid. Left: Spacial domain, Right: Frequency domain. Normalized frequency is $f_0 = \frac{\sqrt{2}}{2^3}$, and orientation 135° .

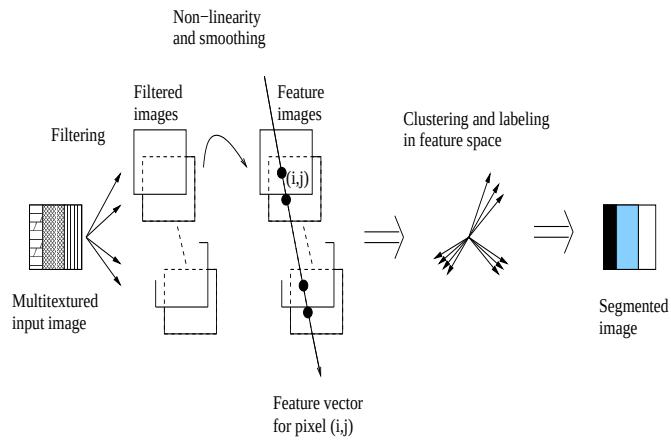
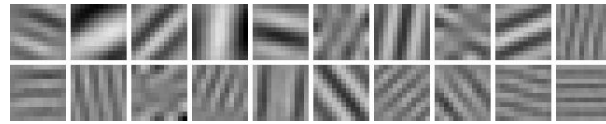
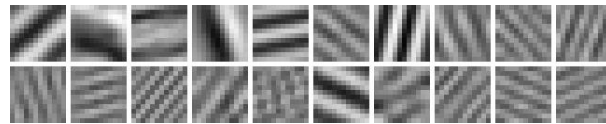


Fig. 7. Experimental setup for segmenting a multi-textured input image.

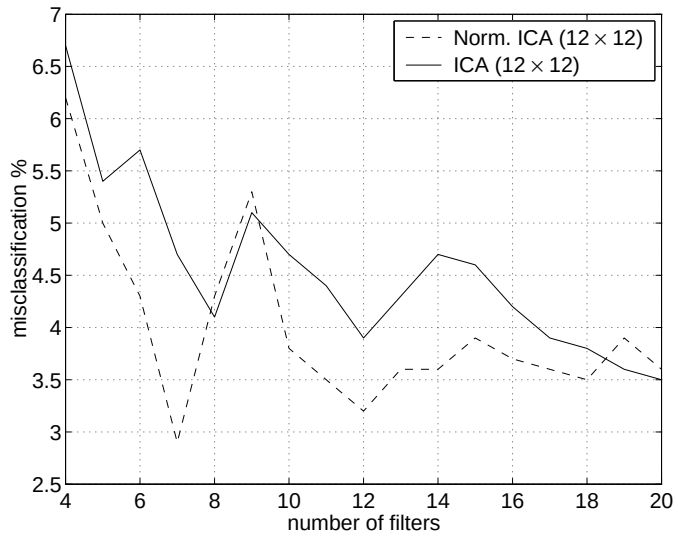


(a)

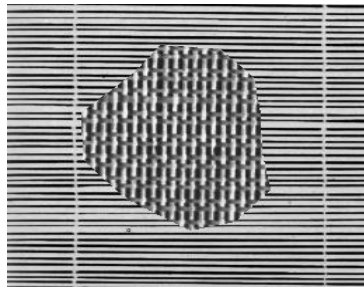


(b)

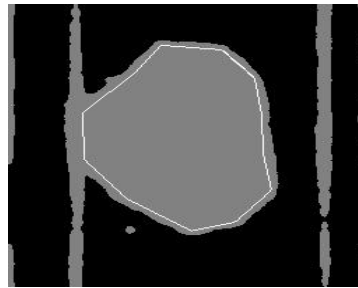
Fig. 8. The 20 most uncorrelated (12×12) ICA functions learned from (a) normalized and (b) non-normalized training textures. The basis functions are ordered from top left to bottom right according to their F -test significance.



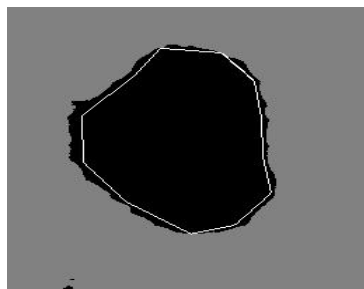
(a) Error vs. number of filters



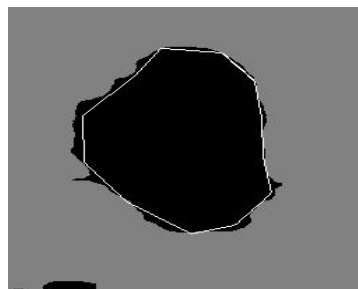
(b) Input image



(c) Gabor, 20. Error 13.8 %

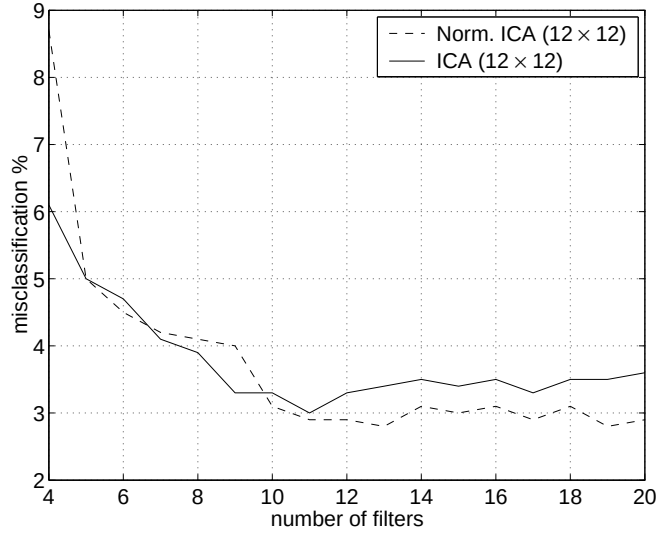


(d) ICA, 20. Error 3.5 %

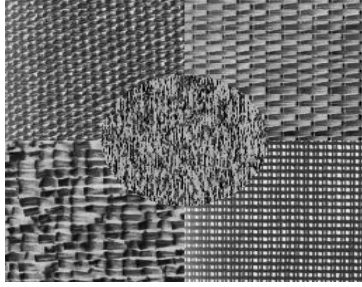


(e) N. ICA, 7. Error 2.9 %

Fig. 9. Segmentation results. Textured image is composed of Brodatz textures $D53$ (inner pattern) and $D49$.



(a) Error vs. number of filters



(b) Input image



(c) Gabor, 20. Error 3.6 %



(d) ICA, 11. Error 3.0 %



(e) N. ICA, 13. Error 2.8 %

Fig. 10. Segmentation results. Textured image is composed of Brodatz textures $D77$, $D55$, $D84$, $D21$ (top left to bottom right) and $D24$ (inner pattern).