ICA FILTER BANK FOR SEGMENTATION OF TEXTURED IMAGES

Robert Jenssen and Torbjørn Eltoft

Department of Physics, University of Tromsø, 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.

1. INTRODUCTION

Independent component analysis [1] has been proposed as a generic statistical model for images [2]. In this case an image is modeled as a weighted sum of basis images. The basic idea in the ICA image model is to construct basis images which for a given image result in weighting components which are mutually statistically independent. The ICA image model is illustrated in figure 1.

In this paper we propose to use the ICA image model 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.

Our ICA based approach to texture segmentation belongs to the so-called filtering methods [3], which include a whole family of algorithms. This approach found motivation in psycho-physiology. Studies suggest that the brain performs a multi-channel frequency and orientation analysis of the visual image formed on the retina [3]. 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 [4].

In the last couple of decades Gabor filters have been widely used in texture segmentation [5]. Gabor filters can be interpreted as oriented band-pass filters [6]. Jain and Farrokhnia [6] used a dyadic even-symmetric Gabor filter bank to generate energy features for texture segmentation, and reported successful results. We use Jain and Farrokhnia's method for comparison with ICA based texture segmentation.

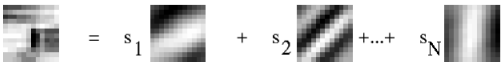


Fig. 1. The ICA image model. An image is modeled as a weighted summation of basis images. The weights are the "independent components".

In the next section we present ICA as a generic statistical model for images. In section 3 we 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.

2. INDEPENDENT COMPONENT ANALYSIS OF IMAGES

In the sequel, we represent an image as a column vector $\mathbf{x} = [x_1, \ldots, x_M]^T$ by re-shaping the image matrix row-by-row into a single column. Mathematically, we express the model as,

$$\mathbf{x} = \sum_{i=1}^{N} s_i \mathbf{a}_i = \mathbf{A}\mathbf{s}, \tag{1}$$

where the basis functions $\mathbf{a}_i$, $i = 1, \ldots, N$, are the columns of the $(M \times N)$ matrix $\mathbf{A}$, and $\mathbf{s} = [s_1, \ldots, s_N]^T$. Treating $\mathbf{x}$ and $\mathbf{s}$ as random vectors, equation (1) defaults to the well-known linear ICA model [1].

In order to learn the matrix $\mathbf{A}$, we attempt to find a linear transformation $\mathbf{W}$ of the training data $\mathbf{x}$, which yields a

vector

$$\mathbf{y} = \mathbf{W}\mathbf{x}, \tag{2}$$

resulting in components of $\mathbf{y}$ being as statistically independent as possible. The vector $\mathbf{y}$ is an estimate of $\mathbf{s}$. The estimate can only be achieved up to a scaling and permutation. Matrix $\mathbf{A}$ is found as the (pseudo) inverse of $\mathbf{W}$.

The training data are considered realizations of random vector $\mathbf{x}$. There exist several iterative algorithms performing ICA. We have used Hyvärinens FastICA algorithm [1]. Usually a PCA preprocessing step is included for whitening of the data.

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. 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.

3. THE ICA REPRESENTATION OF TEXTURE

3.1. Learning methodology

In order to construct ICA basis functions of size $(m \times m)$, the columns $\mathbf{a}_i$, $i = 1, \ldots, N$, of $\mathbf{A}$ in equation (1) 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. Before the selection of training patches, a texture image is rotated a random angle θ, where $\theta \in [0°, 180°]$. 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 [1]. Therefore, without loosing 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 form 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. As a last step, the columns $\mathbf{a}_i$, $i = 1, \ldots, N$, are re-shaped into size $(m \times m)$ to constitute the basis functions. These are now the impulse responses of our filter bank.

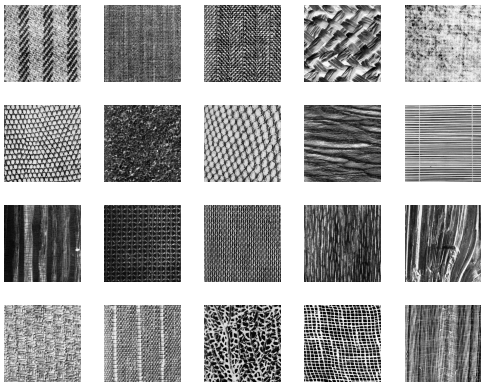
3.2. Characteristics of ICA filter bank

All the textures we use for training the ICA basis functions are taken from the Brodatz album [7]. Figure 2 (a) shows the 20 textures, they are all of size (640×640). Figure 2 (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.

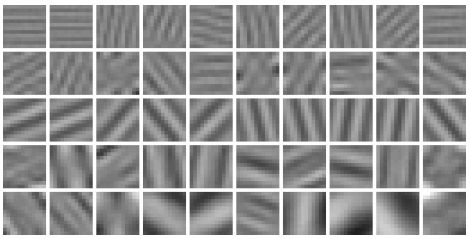
We observe that a majority of the functions are localized in spatial frequency and orientation. That is, they seem to exhibit a kind of sinusoidal wave structure of a specific spatial frequency and to be oriented in a specific direction. Figure 3 (a) shows a scatter plot of the dominating frequency component versus the orientation (in degrees) for the filters of figure 2. 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 2 (b), where function number 1 is to the top left 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 3 shows examples of the frequency domain representation of ICA basis functions number 9 (b) and number 13 (c) of figure 2. We observe that function number 9 has a single frequency component, oriented approximately $45°$ from the horizontal direction. The single dominating



(a) Training textures from the Brodatz album, from top left to bottom right: $D11$, $D16$, $D17$, $D18$, $D19$, $D22$, $D29$, $D35$, $D37$, $D49$, $D50$, $D52$, $D53$, $D68$, $D72$, $D82$, $D85$, $D87$, $D103$ and $D105$.



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

Fig. 2. Training textures and resulting ICA basis functions

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.

3.3. The Gabor filter bank

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

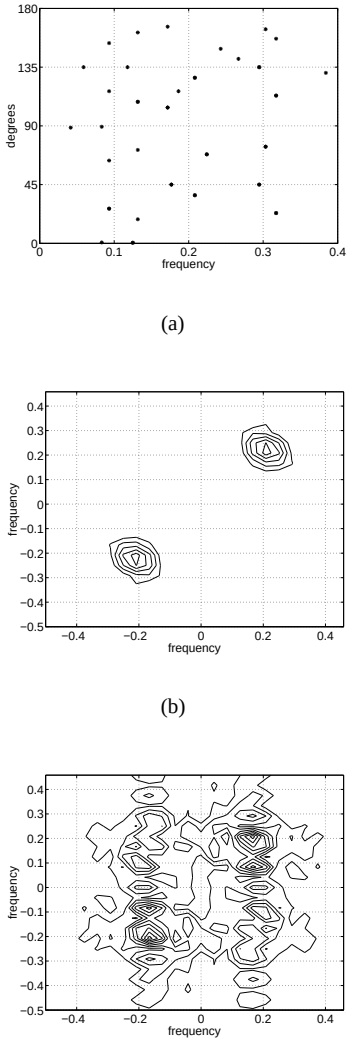
$$h(x,y) = g[q(x,y), w(x,y)] \cos[2\pi f_0 q(x,y)], \quad (3)$$

where

$$g(x,y) = \exp\left\{-\frac{1}{2}\left[\frac{x^2}{\sigma_x^2} + \frac{y^2}{\sigma_y^2}\right]\right\}, \quad (4)$$

$$q(x,y) = \cos(\Phi_0)x + \sin(\Phi_0)y, \quad (5)$$

$$w(x,y) = -\sin(\Phi_0)x + \cos(\Phi_0)y. \quad (6)$$



(a)

(b)

(c)

Fig. 3. Characterization of ICA filters: (a) Scatter plot of dominating frequency component versus orientation (in degrees) for the ICA basis functions of figure 2. (b), (c) Contour plots of frequency domain representation of ICA basis functions number 9 and number 13, respectively

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 space constants of the Gaussian envelope and the spatial frequency are inversely related. We refer to [6] for more details on Gabor filter theory.

The impulse response and frequency content of a typical Gabor filter is shown in figure 4. The frequency selectivity

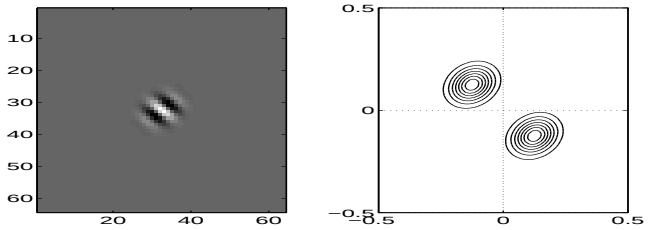


Fig. 4. Example of a Gabor filter in a (64×64) grid. Left: Spacial domain, Right: Frequency domain. Normalized frequency is $f_0 = \frac{\sqrt{2}}{2^3}$, and orientation $135°$.

and orientation properties of Gabor filters are clearly visible.

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 [4],

$$\frac{1\sqrt{2}}{N_c}, \frac{2\sqrt{2}}{N_c}, \frac{4\sqrt{2}}{N_c}, \ldots, \frac{\frac{1}{4}N_c\sqrt{2}}{N_c}. \quad (7)$$

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 (7) are not very useful because they capture features too coarse to represent textural variations in an image, and are therefore often omitted. In our case, for a (256×256) image, the lowest two frequencies of equation (7) are not used. 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) [3], 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. This means that 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 [6].

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 noted from figure 2. The Gabor filters are characterized by their orientation and their center frequency. The ICA filters shown in figure 2 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

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.

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, squaring in our case, 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.

We use a Gaussian smoothing window of size $(l \times l)$ where $l = 2\sigma$ and $\sigma = \frac{2}{\sqrt{3}f_0}$because this choice seems to give the best results in our experiments, both for the ICA filter bank and the Gabor filter bank. For the ICA filters f_0 is an estimated dominating frequency.

As a last step we apply a logarithmic normalizing nonlinearity proposed in [8]. Denoting the k'th filtered image by $r_k(x, y)$, the k'th feature image $e_k(x, y)$ is then given by

$$e_k(x, y) = \log\left(\frac{1}{l^2}\sum_{a,b\in W} r_k^2(x-a, y-b)W(a, b)\right), \quad (8)$$

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

We also need to decide on which, and how many, of the filters displayed in figure 2 to use in the ICA filter bank. To avoid redundancy in 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 [9]. We divide each feature image into R equally sized image regions, $\Omega_r,\ r = 1, \ldots, R$, 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

$$RSS_{kr} = \sum_{i,j \in \Omega_r} [e_k(i,j) - m_{kr}]^2, \qquad (9)$$

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 of all $RSS_{kr},\ r = 1, \ldots, R$.

This value is compared to the RSS_k computed the same way as each RSS_{kr}, but now over the whole feature image $e_k(x,y)$. If RSS_{kT} is close to RSS_k, the feature does not provide local information.

The F-test is based on [9]

$$F_k = \frac{(M-R)\ (RSS_k - RSS_{kT})}{(R-1)\ RSS_{kT}}, \qquad (10)$$

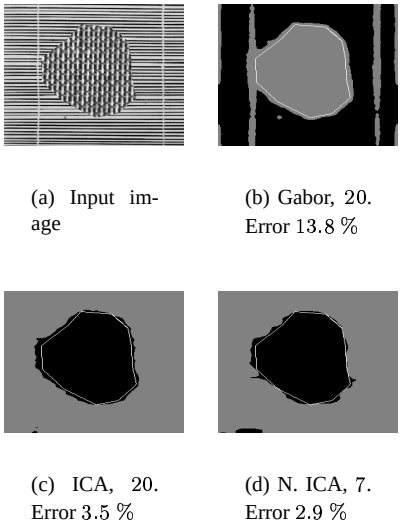
where M is the total number of pixels in each of the regions $\Omega_r,\ r = 1, \ldots, R$.

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.

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 2 (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 is the 20 basis functions with the least frequency overlap, selected among the 50 filters previously shown in figure 2 (b). The second set is obtained in the same manner, only now the training textures were not normalized prior to the ICA basis generation.

The segmentation performance of these two sets are compared to that of a Gabor filter bank consisting of 20 filters. 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. We assume the number of clusters is known apriori, and the initial K prototypes are drawn randomly from the data set. To avoid the problem of the K-means algorithm being stuck in a local minima, we run the algorithm several times, and choose the best clustering according to the K-means cost as our final result.



(a) Input image

(b) Gabor, 20. Error 13.8 %

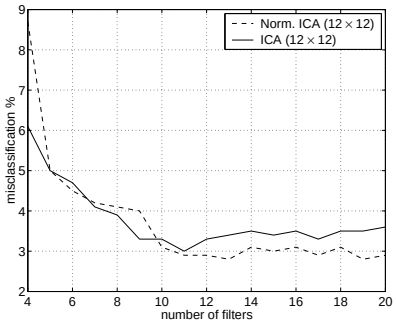
(c) ICA, 20. Error 3.5 %

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

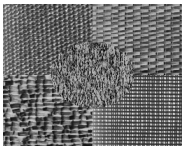
Fig. 5. Segmentation results. Textured image is composed of Brodatz textures $D53$ (inner pattern) and $D49$.

Two multi-textured input images (256×256) are used. Figure 5 (a) shows the first, which is composed of two different texture patterns. The result for the Gabor filter bank is shown in figure 5 (b). It achieves a best error rate of 13.8%. The white solid curve marks the true boundary between the two texture patterns. For the ICA filter banks we only display the best result in each case, for any number of filters, up to maximum 20. The result for the non-normalized case is shown in (c), achieving an error rate of 3.5% using all 20 filters, and the result for the normalized case in (d), with an error rate of 2.9%. We note that the nomalized set performs the best, with an optimum result obtained with only 7 filters. We note that both the ICA sets in this case outperform the Gabor filter bank.

The second multi-textured input image consists of five texture patterns, and is displayed in Figure 6 (b). The performance of an ICA filter bank is obviously depending on how many filters are used. This dependency has been depicted in Figure 6 (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 of their ranking according to the F-test. We note that the first set almost always performs the best, and an optimum result is obtained with 13 filters, resulting in an error percentage of 2.8%. The second set has its best performance using 11 filters, achieving an error percentage of 3.0%. The best result for the Gabor filterbank is in this case 3.6%. The segmented images are displayed in Figures 6 (c), (d) and (e). Also in



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

this case, the ICA filter banks perform better than the Gabor filter bank, allthough the difference is not as large as in the previous experiment.

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.

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. Such filters can be interpreted as oriented bandpass 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.

7. REFERENCES

[1] 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.

[2] A. J. Bell and T. J. Sejnowski, "The "independent components" of natural scenes are edge filters," *Vision Research*, vol. 37, pp. 3327–3338, 1997.

[3] M. Tuceryan and A. K. Jain, *The Handbook of Pattern Recognition and Computer Vision*, chapter 2.1, pp. 207–248, World Scientific Publishing Co., by C. H. Chen, L. F. Pau and P. S. P. Wang (eds.), 2nd edition, 1998.

[4] T. Randen and J. H. Husøy, "Filtering for texture classification: A comparative study," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 21, no. 4, pp. 291–310, 1999.

[5] M. R. Turner, "Texture discrimination by Gabor functions," *Biol. Cybern.*, vol. 55, pp. 71–82, 1986.

[6] A. K. Jain and F. Farrokhnia, "Unsupervised texture segmentation using Gabor filters," *Pattern Recognition*, vol. 24, no. 12, pp. 1167–1186, 1991.

[7] P. Brodatz, "Texture: A photographic album for artists and designers," Dover, New York 1966.

[8] M. Unser and M. Eden, "Nonlinear operators for improving texture segmentation based on features extracted by spatial filtering," *IEEE Transactions on Systems, Man, Cybernetics*, vol. 20, pp. 804–815, 1990.

[9] T. Kasparis, D. Charalampidis, M. Georgiopoulos, and J. Rolland, "Segmentation of textured images based on fractals and image filtering," *Pattern Recognition*, vol. 34, pp. 1963–1973, 2001.