PERGAMON

Pattern Recognition 34 (2001) 1963–1973

PATTERN RECOGNITION
THE JOURNAL OF THE PATTERN RECOGNITION SOCIETY

www.elsevier.com/locate/patcog

Segmentation of textured images based on fractals and image filtering

T. Kasparis[a,]*, D. Charalampidis[a], M. Georgiopoulos[a], J. Rolland[b]

[a]*School of Electrical Engineering and Computer Science, University of Central Florida, Orlando, FL 32816, USA*
[b]*School of Optics/CREOL, University of Central Florida, Orlando, FL 32816, USA*

Received 4 October 1999; accepted 17 August 2000

Abstract

This paper describes a new approach to the segmentation of textured gray-scale images based on image pre-filtering and fractal features. Traditionally, filter bank decomposition methods consider the energy in each band as the textural feature, a parameter that is highly dependent on image intensity. In this paper, we use fractal-based features which depend more on textural characteristics and not intensity information. To reduce the total number of features used in the segmentation, the significance of each feature is examined using a test similar to the *F*-test, and less significant features are not used in the clustering process. The commonly used *K*-means algorithm is extended to an iterative *K*-means by using a variable window size that preserves boundary details. The number of clusters is estimated using an improved hierarchical approach that ignores information extracted around region boundaries. © 2001 Pattern Recognition Society. Published by Elsevier Science Ltd. All rights reserved.

Keywords: Texture segmentation; Gabor filters; Fractal features; Energy features; *K*-means

1. Introduction

Texture is a main characteristic of the surface of an object. In the case of an image, it defines the special relationship between the gray-scale values of the pixels in a region of the image. Texture segmentation is an important task with many applications in pattern recognition and computer vision. Segmentation can be defined as identification of different regions with uniform textures, or as identification of the boundaries between them. For the purpose of segmentation, textures can be described by parameters, usually denoted as features. If the variation of the feature values that describe a uniform textural region is small, then the results obtained by the segmentation process are more accurate. Usually, a feature vector is needed to successfully characterize a textural region. Many features have been used for texture segmentation. Some of them are Gabor energy features [1–3], Fourier transform energy [4], second-order statistical features [5,6] and wavelet features [7,8]. Segmentation involves a method of identification of the different texture regions based on the corresponding features. This method can be clustering or classification of the feature sets that characterize different regions of the image. A commonly used clustering algorithm is the *K*-means. A desired feature property is insensitivity to different image transformations, such as changes in the intensity, zooming, scaling of the gray-scale values and subjection to noise. Fractal dimension (FD) [9–11] has been a popular feature because it is relatively insensitive to changes in the image intensity and to multiplicative noise.

It is well known that the *FD* as a single feature is not sufficient for texture characterization. Chaunduri and Sarkar [9] proposed a solution to this problem by computing the *FD* for different transformations of the original image. Another preprocessing method for producing additional features is based on Gabor filter decomposition where features are computed for each filtered version of the original image. A large number of

* Corresponding author. Tel.: + 1-407-823-5913; fax: + 1-407-823-5835.
E-mail address: kasparis@pegasus.cc.ucf.edu (T. Kasparis).

filters is necessary for efficient texture characterization and as a result a lot of work has been directed towards the reduction of the number of filters by selecting optimal filter parameters. Jain and Farrokhnia [1] proposed a systematic filter selection scheme to select optimal filters from a filter bank that nearly uniformly covers the spatial-frequency domain. Dunn and Higgins [2] use Gabor filters with suitably chosen parameters to detect filter-output discontinuities at texture boundaries. Teuner et al. [3] proposed a method of selecting optimal tuned Gabor filters. The filter parameter selection and tuning are based on the detection of the spectral components of the image. In Refs. [1–3], the segmentation was based on energy features.

In this paper, we extend the work proposed in Refs. [1–3,9] by using fractal features. Specifically, we use a bank of N Gabor filters to filter the original image and the *FD* is computed locally for each one of the filtered images so that the feature vector consisting of N features is obtained. In our experiments, we used 12 filters ($N = 12$). After the features are computed, a clustering algorithm is used for clustering the feature vectors. Each cluster represents one region in the image. Before clustering, the features that do not provide local information are eliminated to speed up the clustering process of the feature vectors. In our work, we also introduce an iterative K-means algorithm that preserves region boundaries better, and provides a cleaner segmentation. Furthermore, we introduce a segmentation approach that does not require the number of regions in the image to be predetermined. This is achieved by using a hierarchical approach for clustering the feature vectors. The proposed iterative K-means clustering algorithm is applied repeatedly assuming each time different number of clusters (regions) in the image. A modification of the VRC index introduced by Calinski and Harabatz [12] is computed for each cluster set and the selected clustering is the one that maximizes this index.

The paper is organized as follows: Section 2, is an overview of Gabor filters and fractal dimension. Section 3 proposes a segmentation technique. Lastly, Section 4 presents the experimental results of segmentation.

2. Background

2.1. Gabor filters and energy

Multi-channel Gabor decomposition is an approach to texture characterization and segmentation [1–3]. The frequency spectrum of a signal, texture in this case, is decomposed into its spectral components using two-dimensional Gabor filters with specified bandwidths and center frequencies. The filters that we use in our work are real-valued directional Gabor filters. These filters can be defined in the spatial and the frequency domain, respectively, as

$$h(x,y) = g(q(x,y), w(x,y)) \cos(2\pi\, uoq(x)), \tag{1}$$

$$H(u,v) = 2\pi\sigma_x\sigma_y\{\mathrm{e}^{(-(q(u,v)-u_o)^2/2\sigma_u^2)-(w(u,v)^2/2\sigma_v^2)} + \mathrm{e}^{(-(q(u,v)+u_o)^2/2\sigma_u^2)-(w(u,v)^2/2\sigma_v^2)}\}, \tag{2}$$

where

$$g(x,y) = \exp\{-[x^2/2\sigma_x^2 + y^2/2\sigma_y^2]\}, \tag{3}$$

$$q(x,y) = \cos(\Phi_o)x + \sin(\Phi_o)y, \tag{4}$$

$$w(x,y) = -\sin(\Phi_o)x + \cos(\Phi_o)y. \tag{5}$$

In the above equations, $g(x,y)$ is the Gaussian envelope, Φ_o denotes the orientation of the filter with respect to u axis, u_o denotes the center frequency of the filter and σ_u, σ_v are equal to $\frac{1}{2}\pi\sigma_x$ and $\frac{1}{2}\pi\sigma_y$, respectively. It is worth noting that $H(u,v)$ is also a Gaussian envelope centered at frequencies $(u_o, 0)$. The parameters σ_x, σ_y are the standard deviations of the Gaussian envelope in the spatial domain and in the direction of $q(x,y)$ and $w(x,y)$, respectively. Hence, σ_x and σ_y define the size of the envelope. The standard deviations of the Gaussian envelope in the frequency domain and in the direction of $q(u,v)$ and $w(u,v)$ are σ_u, σ_v, respectively.

Filtering the original image using filter banks is the first step of feature extraction for texture characterization. The second step is to compute the textural energy in small overlapping windows of the filtered images. Textural energy is a measure that is widely used to characterize texture. The energy that corresponds to a square window of the image Z_f centered at x and y in the spatial domain is defined as

$$E(x,y) = \frac{1}{M^2} \sum_{(i,j)\in W_{xy}} |Z_\mathrm{f}(i,j)|, \tag{6}$$

where $M \times M$ is the size of the window W_{xy}, centered at x, y, and $Z_\mathrm{f}(x,y)$ is the value of the filtered image with coordinates x and y. In Jain et al. [1] the image Z_f is not used directly, but instead a transformation of the form $F(Z_\mathrm{f}(x,y))$, where $F(\cdot)$ is a non-linear, sigmoid function of the form

$$F(t) = \tanh(\alpha t) = \frac{1-\mathrm{e}^{-2\alpha t}}{1+\mathrm{e}^{-2\alpha t}}, \tag{7}$$

where α is a constant.

2.2. Fractal dimension

There are several definitions of the *FD* of an object, including Hausdorff dimension, box dimension and correlation dimension [17]. The *FD* has been characterized as a measure of irregularity of an object. Any curve is an object with one topological dimension that occupies some part of a plane. *FD* defines how much area of this plane is occupied by the curve. For instance, a highly irregular curve will have *FD* greater than that of a straight line. The *FD* of a curve can be between 1, (which is equal to its topological dimension) and 2 (which is equal to the topological dimension of the plane that it can occupy). The concept of *FD* can be extended to surfaces. The *FD* of a surface can be between 2 (which is its topological dimension) and 3 (which is the topological dimension of the "box" that the surface can occupy). Two methods that give good estimation of the *FD* are the differential box counting (DBC) [9] and the variation method [10].

2.2.1. Variation method

The variation method has been adopted in this paper to compute the *FD* of an object because it has been shown to give an accurate and robust estimation of the *FD* of a surface [10]. An image $Z(x,y)$ of size $R \times R$ can be considered as a surface of size $R \times R$, where its value at position (x_o, y_o) is $Z(x_o, y_o)$. According to this method, if a surface Z is a fractal, there exists at least one part of the interval $[0,1]$ where Z is nowhere or almost nowhere differentiable. If $P(x,y,x',y')$ is the slope of the line passing through points $(x,y,Z(x,y))$ and $(x',y',Z(x',y'))$, then $|P(x,y,x',y')|$ goes to infinity as the point (x',y') tends toward (x,y). The *FD* is defined as the rate in which $|P(x,y,x',y')|$ goes to infinity. The variation of Z can be defined as

$$V_\varepsilon(x,y = \max_{\mathrm{dist}((x,y),(s,t))\leqslant\varepsilon} Z(s,t) - \max_{\mathrm{dist}((x,y),(s,t))\leqslant\varepsilon} Z(s,t), \tag{8}$$

where $\mathrm{dist}((x,y),(s,t)) = \max(|x-s|,|y-t|)$ and $\varepsilon > 0$. The integral of $V_\varepsilon(x,y)$ tends to zero as ε tends to 0. The rate of growth of this integral is directly related to the *FD* of Z. The *FD* of the surface Z is then defined as

$$FD_z = \Delta_V(Z) = \lim_{\varepsilon\to 0} \frac{\log \int_0^1\int_0^1 (V_z(x,y)/\varepsilon^3)\,\mathrm{d}x\,\mathrm{d}y}{\log\frac{1}{\varepsilon}}. \tag{9}$$

The slope of the log–log plot of the line that is defined by $\log\iint[V_\varepsilon(x,y)/\varepsilon^3]\,\mathrm{d}x\,\mathrm{d}y$ and $\log(1/\varepsilon)$ gives the *FD* of the surface. The computation of the *FD* of a discretized surface involves the substitution of the integrals with summations.

The *FD* of an image can be computed locally in all the different regions of size $R \times R$ of the image, so that a *FD* space can be created. This *FD* space will be mapped one-to-one to the pixels of the image. The algorithm for computing the *FD* space of an image is implemented as follows: The difference V_ε between the maximum and the minimum gray-scale values is computed in a small window of size $T \times T$, where $T = 2\varepsilon + 1$. This window is centered at the pixel with coordinates (x,y). This computation is repeated for all pixels (x,y) of the image, for $\varepsilon = 1,2,3,\ldots,\varepsilon_{\max}$. $V_\varepsilon(x,y)$ is the εth variation located at (x,y). If we define E_ε as the average of $V_\varepsilon(x,y)$ over a window W of size $R \times R$, then the *FD* located at the window W is the slope of the line that best fits the points $(\log(R/\varepsilon), \log\{(R/\varepsilon)^3 E_\varepsilon\})$, where $\varepsilon = 1,2,3,\ldots,\varepsilon_{\max}$. The line that best fits these points can be found using linear regression. The *FD* is then mapped to the central pixel of the window *W*. The next step is to shift the window *W* and map the *FD* to the central pixel of the new window. The previous steps are repeated for all pixels of the image and the *FD* space is thus created.

3. A combined segmentation method

The *FD* as a single feature is not sufficient for texture analysis and characterization. The idea of using more than one *FD* feature has already been introduced [9,10]. In this paper, a feature set consisting of the *FD* of 12 filtered versions of the original image is used. Directional, non-symmetric, real-valued, two-dimensional Gabor filters are used for the filtering. The idea employed is to use the *FD* instead of the energy of the filtered images, because the *FD* is insensitive to differences in the local intensity of the image and to local scaling of the gray-scale levels. Ideally, the *FD* is also insensitive to image zooming but in practice this is only valid to a certain extent. The segmentation method that is described in this paper consists of two steps: feature extraction and clustering of feature vectors.

3.1. Feature extraction

The first step in feature extraction is filtering of the original image Z_o using Gabor filters with different center frequencies and orientations. The filters are $H(q,f,\sigma_x,\sigma_y)$, where q defines the orientation of the filter, f defines the center frequency and σ_x, σ_y are the standard deviations of the Gaussian envelope in the spatial domain. The second step is to compute the *FD* for all filtered versions of the image. The *FD* values that are computed over a window W centered at the pixel with coordinates (x,y) for all filtered versions of the image, constitute the image feature vector at coordinates (x,y). These *FD* values comprise the feature set. The standard deviations used are $\sigma_x = 6$ in the direction of the filter and $\sigma_y = 0.01$ in the direction which is perpendicular to the direction of the filter. These filters decompose the image into its frequency components essentially in the direction of the filters. Other types of filters were also tried but the percentage of correct clustering was experimentally found to be higher when this filter type was used. Three normalized center filter frequencies $f = 0, 0.05$ and 0.1 were used. One reason that leads to the selection of lower center frequencies for the Gabor filters,

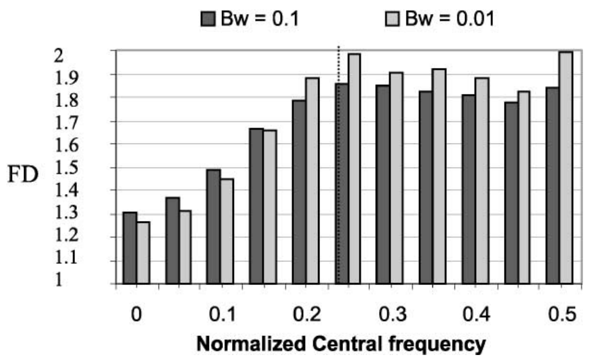

Fig. 1. *FD* versus the center frequency of bands with bandwidths 0.01 and 0.1.

is that images contain most information at lower frequencies.

In our experiments, we used four filter orientations $q = 0, 45, 90, 135°$. Four orientations were found to be sufficient for the filters to detect the directional characteristics of the textures. Three different center frequencies and four different orientations define a set of 12 filters that are used to filter the original image.

One indication why the specific filters are used, is given by the relation between *FD* and frequency that is shown in Fig. 1. The *FD* was computed for one-dimensional signals that contain only a specific band of frequencies, with center frequencies $f_c = 0.05k$ where $k = 0, 1, \ldots, 10$, and bandwidths $Bw = 0.1$ and 0.01, all in normalized units. From Fig. 1, we notice that the *FD* increases as the center frequency increases up to 0.25 and fluctuates thereafter. This result leads us again to use relatively low central filter frequencies ($f_c < 0.2$).

If the Gabor filters that we used in our experiments isolated narrow frequency bands in all directions, the *FD* would have an approximately specific value according to Fig. 1. For this reason, the filters we are using isolate a relatively small frequency band only in one direction. The result in Fig. 1 does not imply that the *FD* depends only on frequency — it shows that the *FD* is related to frequency, and it gives an indication why the filters should not be narrowband in all directions.

For the computation of the *FD*, the slope of the line that passes through two points $(\log(R/\varepsilon), \log\{(R/\varepsilon)^3 E_\varepsilon\}), \varepsilon = (1, 2)$ is considered. The reason for selecting only two points is that the main interest is not the exact value of the *FD*, but the robustness of the feature and its power to discriminate between different textures. Larger values of ε involve larger windows for computing the variations. The larger the window is, the higher the probability of including regions of more than one texture. We used windows of size $R \times R$ with $R = 16$. A larger window size gives a more robust feature set, but at the same time it blurs the boundaries between textures. Also, larger windows increase the sensitivity of *FD* to image intensity.

There are certain advantages of using *FD* over other features. One advantage is that the *FD* is independent of the overall intensity of the image [14–16] meaning that the *FD* is invariant to a constant offset *C*. Practically, the offset value *C* could be slowly varying since it is sufficient to be almost constant within a window of size $(2\varepsilon + 1) \times (2\varepsilon + 1)$. The *FD* is also insensitive to multiplicative noise. The proof is evident considering that if the gray-scale values in a region around a pixel with coordinates (x, y) are multiplied by a constant factor *M*, the new maximum and minimum values are simply multiplied by the same constant *M*. If the average variation around (x, y) was $E_\varepsilon^{(\mathrm{old})}$, then the new average variation is $E_\varepsilon^{(\mathrm{new})} = M\, E_\varepsilon^{(\mathrm{old})}$. The *FD* is equal to the slope of the line that better fits the points $(\log(R/\varepsilon), \log\{(R/\varepsilon)^3 M\, E_\varepsilon\})$ or $(\log(R/\varepsilon),\ \log\{(R/\varepsilon)^3 E_\varepsilon\} + \log M)$. The presence of the constant term $\log M$ does not change the slope of the line, thus the *FD* remains unchanged. Practically, *M* can slowly vary because it is sufficient to be fairly constant in a relatively small window (like 16×16). This implies that if different regions of the image are multiplied by different constant factors, the segmentation results will not change significantly. In contrast, energy is sensitive to multiplicative noise, since the output Z_f of a filter is the convolution $Z_f(x, y) = h(x, y) * Z_o(x, y)$, where Z_o is the original image and $h(\cdot)$ is the two-dimensional impulse response of the Gabor filter. If the gray-scale values of the original image are multiplied by a constant factor *M*, then the filtered image is also scaled by *M*. Therefore, the energy will change and it is possible that an otherwise homogeneous region will be segmented incorrectly.

3.2. Clustering of the feature vectors

3.2.1. Feature reduction

All 12 *FD* features that are mapped to the pixel with coordinates (x, y) form a feature vector. There is a total of $S \times S$ feature vectors of size 12, where *S* is the width of the image.

Eliminating features that possess small variability over the textured image can reduce the size of the feature vectors. Let us define the residual sum of squares (*RSS*) as

$$RSS = \sum_i (x_i - m)^2, \tag{10}$$

where x_i are the values of the feature in this region and m is the mean of the feature at this region. One way of measuring the variability of a feature is to separate the image in smaller, equally sized regions. Then, we compute the *RSS* in each one of the regions and the sum of all the calculated *RSSs*. The sum represents the total *RSS* and will be denoted as RSS_T. The RSS_T is then compared to the *RSS* of the feature for the whole image, which we denote as RSS_o. If RSS_T is close to RSS_o of the whole image, then we know that the feature does not provide local information.

The measure F we used to identify significant features is similar to the F-test used in statistics which is defined as

$$F = \frac{(n-k)(RSS_o - RSS_T)}{(k-1)RSS_T}, \tag{11}$$

where n is the total number of pixels, and k is the number of regions in the image. Values of F above a threshold U_T indicate that the feature is significant, whereas smaller values indicate non-significant features. The threshold U_T is determined from the feature corresponding to the largest F value.

3.2.2. Iterative K-means clustering

A K-means-based algorithm was used to cluster the feature vectors. This algorithm considers the boundaries between textures so that a finer segmentation at the boundaries is possible. For better segmentation, the features are smoothed over a window. The algorithm is summarized into the following steps:

Step 0: Set the square window length equal to $R = 33$, and give to all pixels the label A.

Step 1: Smooth the features of the image by averaging their values in a square window of size $R \times R$ and map the result to its central pixel.

Step 2: Apply the original K-means algorithm for the pixels with label A.

Step 3: A sliding window of size $R \times R$ merges small regions to large regions. This window classifies the feature vector that is mapped to its central pixel to class j, if the number of feature vectors that are associated to class j, is larger than the number of features that are associated to any other class.

Step 4: Re-examine all pixels:

(a) If the pixel is an unambiguous pixel give it the label U.
(b) If the pixel is an ambiguous pixel give it the label A.

Step 5: Examine two cases:

(a) If your previous window length is R where $R > 9$ reduce it to $R = R - 8$, and go to step 1.
(b) If your previous window length is 17 stop.

Consider a window $R \times R$ centered at a designated pixel. This pixel is defined as ambiguous if any other pixel in this window has a different assigned classification. The labels A and U do not indicate the cluster in which the pixel is clustered. They only indicate the pixel status, i.e. they suggest if the pixel should be re-examined (label A) or not (label U). We must also note that in the cases where the smoothing or merging windows are centered at a pixel close to the edges of the image, these windows may exceed beyond the image limits. In that case, we assume that the image continues beyond the edges, and it is taken to be equal to the mirrored version of the original image.

One disadvantage of the K-means algorithm is that the random selection of centers can lead to a local minimum. Applying the algorithm more than once for different selections of the initial centers increases the probability to reach the global minimum. The criterion for determining which clustering is better is the minimization of the square error, which is equivalent to minimizing the quantity

$$SE = \sum_{i=1}^{f} \sum_{j=1}^{n} (x_{ij} - m_{ij})^2, \tag{12}$$

where x_{ij} is the value of the ith feature of the jth feature vector, and m_{ij} is the corresponding center (mean) of the ith feature that has been computed for all feature vectors that belong to the same cluster as the jth feature vector. In Eq. (12), f is the size of the feature vector.

3.2.3. Modified VRC index

Another disadvantage of the K-means algorithm is that the number of clusters has to be predefined. Generally, estimation of the number of clusters is a difficult task. In our work, we used a hierarchical approach based on the index *VRC* that has been introduced by Calinski and Harabatz [12] to select the number of clusters. Milligan and Cooper [13] have classified this index as the first, among 30 indices. The clustering algorithm presented in Section 3.2.2 is applied for two clusters, three clusters and so on. In our experiments we tried up to nine clusters. The index is given by

$$VRC = \frac{(n-k)BCSS}{(k-1)WCSS}, \tag{13}$$

where *BCSS* is the "between-clusters sum of squares" and *WCSS* is the "within-cluster sum of squares", n is the total number of pixels, and k is the number of clusters. The within-cluster sum of squares is defined as

$$WCSS = \sum_{i=1}^{f} \sum_{j=1}^{n} (x_{ij} - m_{ij})^2, \tag{14}$$

where x_{ij} is the value of the ith element of the jth feature vector, and m_{ij} is the corresponding center. We notice that *WCSS* is equivalent to the square error as it has been defined in Eq. (12). For the definition of *BCSS*, it is useful to define the total sum of squares

$$TSS = \sum_{i=1}^{f} \sum_{j=1}^{n} (x_{ij} - m_i)^2, \tag{15}$$

where x_{ij} is the value of the ith element of the jth feature vector, and m_i is the corresponding mean of the feature through the whole image. Thus, *BCSS* is defined as

$$BCSS = TSS - WCSS \tag{16}$$

The selection of the number of clusters is based on the maximization of the index *VRC*.

Generally, the feature values at the boundaries are not the representative ones of the cluster in which they belong. For this reason, we modified the index so that the feature values at the boundaries will not be included. The new index is of the form

$$VRC_m = \frac{(n-k)BCSS_m}{(k-1)WCSS_m} \tag{17}$$

where $WCSS_m$ is given by Eq. (14), without including the terms that correspond to pixels that are close to boundaries. Similarly,

$$BCSS_m = TSS - \frac{n}{n_{used}} WCSS_m \tag{18}$$

where n is the total number of pixels and n_{used} is the number of pixels that are not close to the boundaries. The normalization term n/n_{used} is used because $WCSS_m$ does not include all pixels for the computation of the within cluster sum of squares. The index was calculated by not considering pixels that are closer than 6, 8 and 10 pixels to the estimated boundaries.

4. Experimental results

The segmentation algorithm that is presented in this paper is automatic in the sense that the number of textures in the image does not have to be predefined. Also, the number of features that are used varies, and features that do not provide any useful information are eliminated. Even though the algorithm is automatic, in the experiments described in Sections 4.1–4.3 the number of *FD* features is fixed to 12 and the number of textures is also forced to be the correct one. The reason is that we want to compare the segmentation performance between the *FD* and energy as well as between the original and the iterative *K*-means, with respect to the percentage of correct clustering (*PCC*). In Section 4.4, we examine the performance of the full algorithm. We test the performance of the modification of the index *VRC* that was presented in Section 3.2. The test images used in the experiment were mosaics of texture samples from Brodatz [18].

4.1. Comparison between symmetric and non-symmetric filters for FD

In this work, we use non-symmetric Gabor filters because experiments have shown better performance when the *FD* feature is used. Fig. 2 presents a comparison. Fig. 2(a) is the original mosaic and Fig. 2(b) is the segmentation using non-symmetric filters (as described in Section 3.1). For a comparison, Fig. 2(c) and (d) present the segmentation using symmetric filters, where it can be seen that the performance with the non-symmetric filters is better.

4.2. Comparison between the FD and energy features

Three types of experiments were performed using two different mosaics. In the first type, the approach of Jain and Farrokhnia [1] was followed, where symmetric Gabor filters and energy are used. In the second and third types of experiments we used non-symmetric filters and either *FD* or energy as features. Fig. 3(a) presents an original mosaic, Fig. 3(b) presents the segmentation using the proposed method, and Fig. 3(c) is the segmentation following the approach of Jain and Farrokhnia. It can be observed that segmentation using the *FD* is slightly better. In Fig. 3(b) the *PCC* is 93% and in Fig. 3(c) the *PCC* is 88%. Fig. 4(a)–(c) compare segmentation results using energy features when non-symmetric filters are used. The *PCC* in Fig. 4(b) with *FD* features is 93% and in Fig. 4(c) with energy features is 91%.

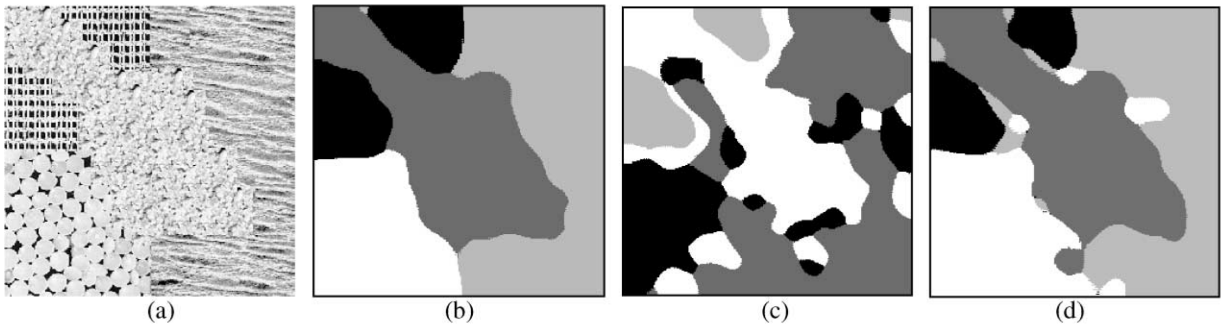

Fig. 2. Segmentation using *FD*: (a) original image, (b) using non-symmetric filters ($\sigma_x = 6$, $\sigma_y = 0.01$), (c) using symmetric filters ($\sigma_x = 6$, $\sigma_y = 6$), (d) using symmetric filters ($\sigma_x = 3$, $\sigma_y = 3$).

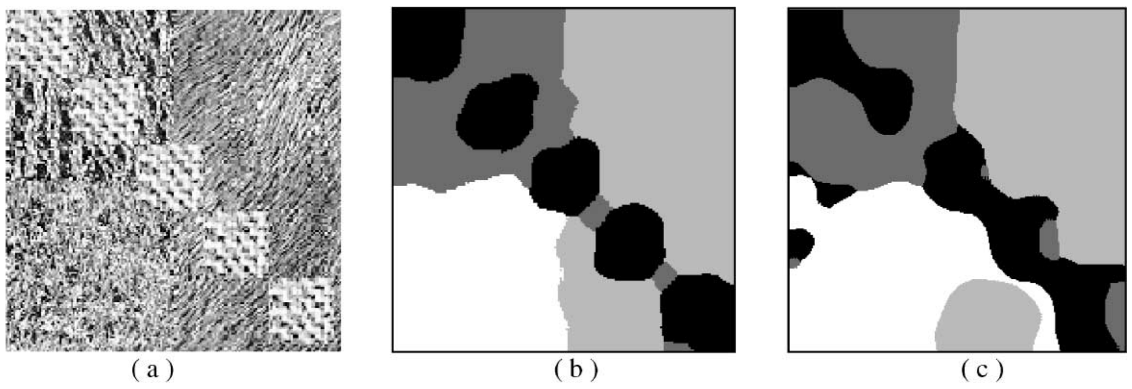

Fig. 3. Segmentation: (a) original image, (b) using *FD* and non-symmetric filters, (c) using symmetric filters and energy.

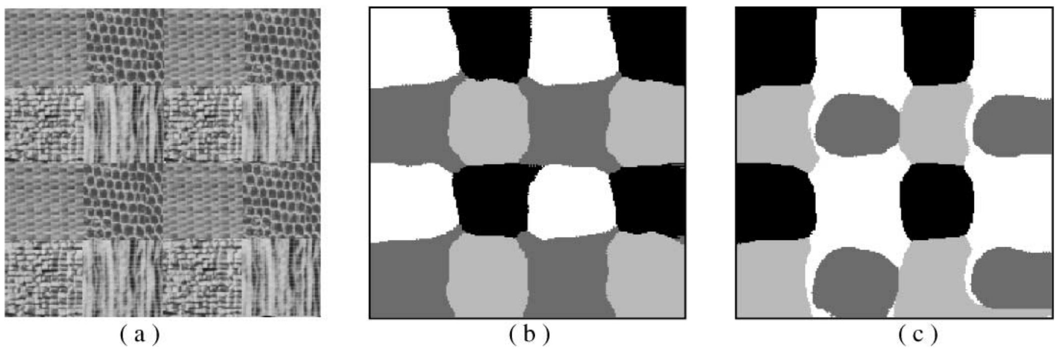

Fig. 4. Segmentation: (a) original image, (b) using *FD*, (c) using non-symmetric filters and energy.

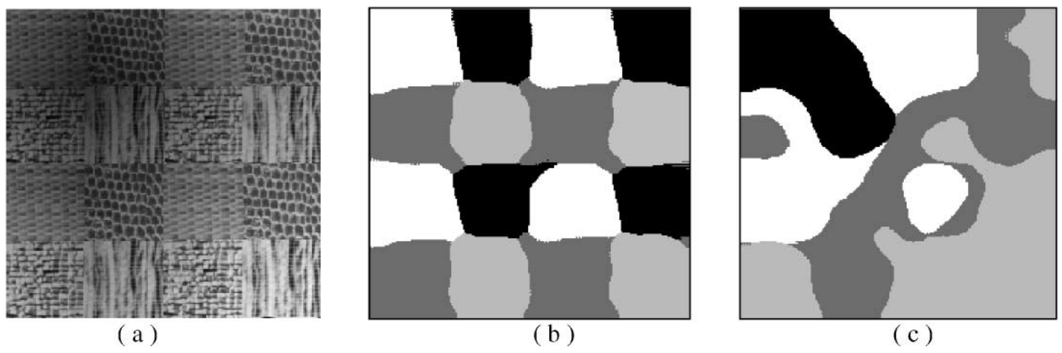

Fig. 5. Segmentation: (a) image affected by multiplicative noise, (b) using *FD*, (f) using non-symmetric filters and energy.

Fig. 5 demonstrates the robustness of the *FD* under multiplicative noise. Fig. 5(a) is the original of Fig. 4(a), corrupted by multiplicative noise. Fig. 5(b) shows the segmentation using *FD* and Fig. 5(c) shows the segmentation using energy. It can be observed that the performance using *FD* remains virtually the same as with the noise-free case, whereas the segmentation using energy degrades drastically. The *PCC* for Fig. 5(b) is 92.6%, virtually the same as the *PCC* of the noise-free image in Fig. 4(b).

4.3. Comparison between standard and iterative K-means

The standard *K*-means algorithm is compared to the proposed iterative *K*-means algorithm. For the standard, two sizes of the smoothing window were chosen: a small window of size 17×17 and a large window of size 33×33. When the standard *K*-means is used with large feature smoothing and merging windows, irregular boundaries are smoothed out. On the other hand, if a smaller window is used, small isolated regions may be produced

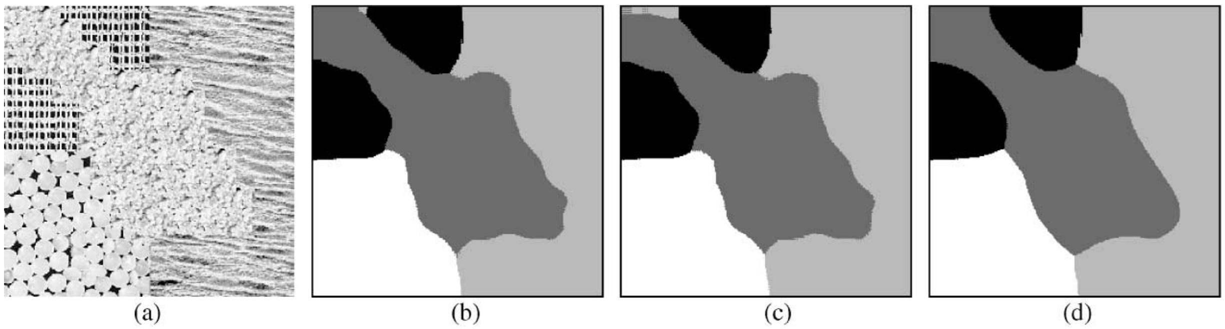

Fig. 6. Segmentation using iterative and standard *K*-means for *FD*. Images: (a) original, (b) segmented by iterative *K*-means, (c) segmented by standard *K*-means and smoothing window of size 17×17, (d) segmented by standard *K*-means and smoothing window of size 33×33.

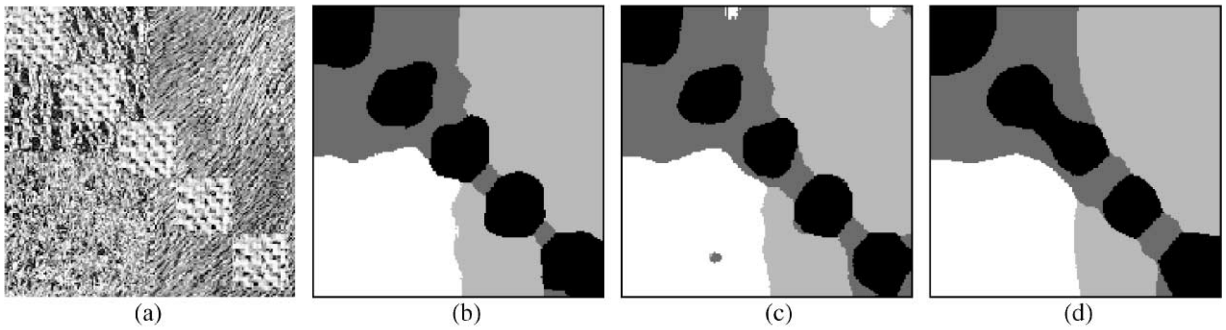

Fig. 7. Segmentation using iterative and standard *K*-means for *FD*. Images: (a) original, (b) segmented by iterative *K*-means, (c) segmented by standard *K*-means and smoothing window of size 17×17, (d) segmented by standard *K*-means and smoothing window of size 33×33.

within otherwise homogeneous regions. The proposed iterative *K*-means minimizes these problems by starting the iterations with a large window and gradually reducing the size.

Fig. 6 compares segmentation results using the iterative and standard *K*-means. Fig. 6(a) is the original image, and Fig. 6(b) is the segmentation using the iterative *K*-means. Fig. 6(c) presents segmentation using the standard *K*-means with a smoothing window of size 17×17, where it can be noticed that there is some misclassification at the upper left corner of the image. In Fig. 6(d) the standard *K*-means with a smoothing window of size 33×33 was applied. Here, it can be seen that the boundaries are excessively smoothed. Comparing Fig 6(b)–(d), we can conclude that the iterative *K*-means produced better segmentation. Fig. 7 is another segmentation example where similar comments apply. The *PCC* was 94.4% for the iterative *K*-means and around 91% for the standard *K*-means.

4.4. Feature reduction

For the segmentation experiments described in Sections 4.1–4.3, we used all 12 *FD* features for the segmentation. In Section 3.2.1, we described an approach of eliminating some of the less significant features by applying a threshold U_T, which was selected equal to $F_{\max}/4.5$, where $F_{\max}$ is the maximum F value over all other features. This approach was tested for a number of textured images, and the results using less features were almost identical to the ones where all 12 features were used (the change in *PCC* was less than 1%). Elimination of some of the features is important because the method that is used for estimating the number of clusters is hierarchical which is generally time-consuming, and elimination of some of the features reduces computation. For our experiments, the average number of features that were used was 9.95. In Fig. 8, an example is presented. Fig. 8(a) is the original image, Fig. 8(b) is the segmented image where 12 features are used, and Fig. 8(c) is the segmented image where 8 features above the U_T threshold were used. From Fig. 8, we can conclude that 12 features and 8 features result in almost identical segmentations.

A rough calculation of the computational complexity of our technique shows that the feature extraction algorithm is of the order of $O(N_p N_F)$ where N_p is the

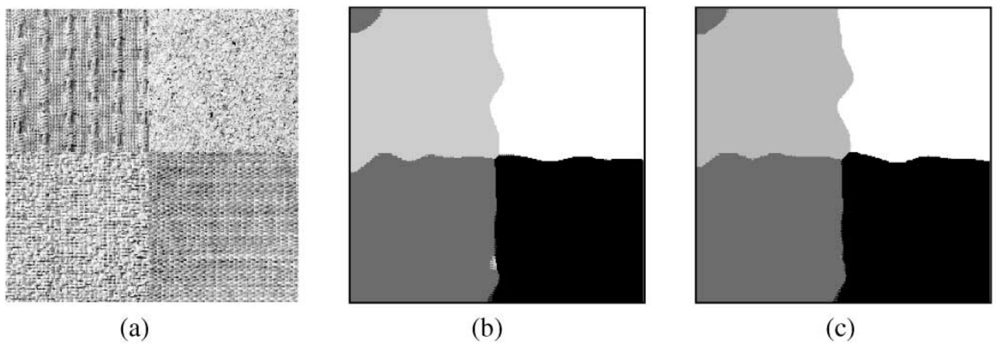
Fig. 8. Segmentation: (a) original image, (b) segmented using 12 features, (c) segmented using 8 features.

number of pixels in the image, and N_F is the number of features associated with each pixel. The clustering algorithm, which is the most time consuming step, is of the order of $O((N_{MC})^2 + 4N_{MC}N_{FR})$, where N_{MC} is the maximum number of clusters assumed, and N_{FR} is the number of features associated to a pixel, after feature reduction.

4.5. Estimating the number of textures in the image

In Section 3.2.3, we described an approach for estimating the number of clusters in an image. The index *VRC*, as well as the modified index VRC_m (see Section 3.2.3) where pixels close to the boundaries were not included in the estimation of the number of clusters, were tested. In our experiments, we tested four cases. In the first case, all pixels in the image were used, and in the other three, pixels that are closer than 6, 8 and 10 pixels to the boundaries are not included in the computation of the index. The indices are denoted as VRC, VRC_m^1, VRC_m^2 and VRC_m^3, respectively. We tested the performance of the four indices by estimating the number of clusters in 20 different images.

Table 1 presents the results of the experiments, where we notice an improvement when the pixels that are close to the boundaries are not included. The maximum number of correct cluster estimations is given by the index VRC_m^2 where pixels that are closer to the boundaries than 8 pixels are not included. From the results presented in Table 1, we can conclude that ignoring some of the feature vectors that are close to the boundaries can improve the estimation since these vectors are not the most representative ones of the cluster that they are associated with. On the other hand, ignoring more vectors leads to omitting useful information, especially for small clusters.

Table 1
Number of correct and incorrect cluster estimations

	Correct estimation	Incorrect by one cluster	Incorrect by 2 clusters	Incorrect by more than 2 clusters
VRC	13	5	1	1
VRC_m^1	14	4	1	1
VRC_m^2	15	4	1	0
VRC_m^3	14	5	1	0

5. Conclusions

In this paper, we extended previous work on Gabor filter-based texture segmentation by using a feature vector that consists of fractal-based features. Gabor filter decomposition of images offers a solution to the problem of finding a large number of features. Since the number of filter banks can be adjusted, texture characterization can be improved by increasing the number of filters in the case of difficult segmentation problems. It has been shown in the literature that *FD* is insensitive to linear transformations of image intensity. Therefore, *FD* can be used for real applications where images have been obtained in different or badly illuminated environment. On the other hand, one or a few *FD* features may not be enough for successful segmentation. In the proposed feature extraction scheme, filter decomposition allows the important properties of *FD* to be of use, since a large number of *FD* features can be produced in a systematical manner. In our experiments, we found that the proposed *FD* vector has certain advantages over traditional energy measures since it is insensitive to changes in the image intensity and to multiplicative noise.

Experiments have shown that when boundary regions are irregular, the iterative *K*-means is superior to the standard *K*-means algorithm. The iterative *K*-means preserves nicely the textural boundaries in most segmentation cases. The feature vectors are initially extracted in relatively small windows of the image and then they are smoothed using different window sizes depending on how close we are to the boundary between the estimated textural regions. There is an advantage in using our technique over the technique where features are extracted in windows of variable size. Feature extraction is generally a time-consuming process compared to smoothing in that it is not efficient to extract new features each time a boundary region needs to be further refined.

The previously proposed Calinski and Harabatz index that is used for the estimation of the number of clusters was modified so that the feature vectors that are close to the boundaries are not used for the estimation of the number of textures. Generally, features extracted from texture boundaries represent a mixture of textural characteristics of the different regions. If the area of boundary regions to be ignored is large, small textural regions will also be ignored. Hence, the number of regions in the image will be underestimated. It is important to note that in our experiments we found that the number of textural regions is satisfactorily estimated when the areas closer than 8 pixels to the texture boundaries are ignored. This is half the width of the smallest smoothing window that is used in the iterative K-means algorithm. If a smoothing window of size 16×16 is at least 8 pixels away from the boundary it includes a part of only one textural region. Therefore, if we consider only pixels far away from the boundaries by at least 8 pixels, then the smoothing window represents information extracted from a pure textural region.

6. Summary

We have proposed a new approach for the segmentation of textured gray-scale images based on image filtering and fractal features. The original image is filtered using 12 different Gabor filters of four orientations and three different normalized center frequencies. Some justification on the specific choice of filters has been presented. The fractal dimension extracted from the filtered images is used as one of the features that are computed within a window and mapped to the central pixel. A method of eliminating insignificant features is also proposed. An iterative K-means-based algorithm which includes feature smoothing and takes into consideration boundaries is used to segment an image into a number of clusters. The iterations begin with a large window size and it is gradually reduced during each iteration. The number of clusters is estimated using a hierarchical approach and a modified index that improves the estimation of the number textures in an image. Experimental comparisons show that the fractal features outperform the commonly used energy-based features, especially under certain types of image distortion. The iterative K-means is found to produce more precise segmentation when the textural boundary shapes are irregular.

References

[1] A.K. Jain, F. Farrokhnia, Unsupervised texture segmentation using Gabor filters, Pattern Recognition 24 (1991) 1167–1185.

[2] D. Dunn, W.E. Higgins, Optimal Gabor filters for texture segmentation, IEEE Trans. Image Process. 4 (1995) 947–964.

[3] A. Teuner, O. Pichler, B.J. Hosticka, Unsupervised texture segmentation of images using tuned matched Gabor filters, IEEE Trans. Image Process. 4 (1995) 863–870.

[4] R. Bajscy, Computer identification of visual surfaces, Comput. Graphics Image Process. 2 (1973) 118–130.

[5] P.C. Chen, T. Pavlidis, Segmentation of texture using correlation, IEEE Trans. Pattern Anal. Mach. Intell. 5 (1983) 64–69.

[6] V. Murino, C. Ottonello, S. Pagnan, Noisy texture classification: a higher — order statistics approach, Pattern Recognition 31 (1998) 383–393.

[7] M. Unser, Texture classification and segmentation using wavelet frames, IEEE Trans. Image Process. 4 (1995) 1549–1560.

[8] R.N. Strickland, Wavelet transform methods for image detection and recovery, IEEE Trans. Image Process. 6 (1997) 724–734.

[9] B.B. Chaundhuri, N. Sarkar, Texture segmentation using fractal dimension, IEEE Trans. Pattern Anal. Mach. Intell. 17 (1995) 72–77.

[10] B. Dubuc, C. Roques-Carmes, C. Tricot, S.W. Zucker, The variation method: a technique to estimate the fractal dimension of surfaces, SPIE Visual Commun. Image Process. II 845 (1987) 241–248.

[11] T. Kasparis, N.S. Tzannes, M. Bassiouni, Q. Chen, Texture description based on fractal and energy features, Comput. Electrical Enging 21 (1995) 21–32.

[12] T. Calinski, J. Harabatz, A dendrite method for cluster analysis, Commun. Stat. (1974) 1–26.

[13] G.W. Milligan, M.C. Cooper, An examination of procedures for determining the number of clusters in a data set, Phychometrica (1985) 159–179.

[14] D. Charalampidis, T. Kasparis, J. Rolland, Segmentation of textured images based on fractal feature combinations, Proc. SPIE, Visual Inform. Process. VII (1998) 25–35.

[15] A. Pentland, Fractal-based description of natural scenes, IEEE Trans. Pattern Anal. Mach. Intell. 6 (1984) 661–674.

[16] J. Garding, Properties of fractal intensity surfaces, Pattern Recognition Lett. 8 (1988) 319–324.

[17] Y.B. Pesin, Dimension Theory in Dynamical Systems, Contemporary Views and Applications, The University of Chicago Press, Chicago, 1998.

[18] P. Brodatz, Texture: A Photographic Album for Artists and Designers, Dover, New York, 1966.

About the Author—TAKIS KASPARIS received the Diploma of Electrical Engineering from the National Technical University of Athens-Greece in 1980, and the MEEE and Ph.D. degrees in Electrical Engineering from the City College of New York in 1982 and 1988. From 1985 until 1989 he was also an electronics consultant. In 1989 he joined the Electrical Engineering Department of the University of Central Florida, Orlando, where he is presently an associate professor of EE within the School of Electrical Engineering and Computer Science. His research interests are in digital signal and image processing, with more emphasis in texture analysis, non-linear filtering, and processing of remote sensing data.

About the Author—DIMITRIOS CHARALAMPIDIS received the Diploma of Electrical Engineering from the university of Patras, Greece in 1996 and the MS Degree in Electrical Engineering from the University of Central Florida in 1998. He is currently completing the Ph.D. degree at the current institution in the area of image processing. His research interests include image processing, pattern recognition and neural networks.

About the Author—MICHAEL GEORGIOPOULOS received the Diploma of Electrical Engineering from the National Technical University of Athens–Greece in 1981, and the MEEE and Ph.D. degrees in Electrical Engineering from the University of Connecticut, Storrs, in 1983 and 1986. Afterwards he joined the Electrical Engineering Department of the University of Central Florida, Orlando, where he is presently an associate professor of EE within the School of Electrical Engineering and Computer Science. His research interests are in Neural Networks, Pattern Recognition, Smart Antennas, and Applications of Neural Networks in Communications, Human Behavioral Modeling and Data Mining.

About the Author—JANNICK P. ROLLAND is an Assistant Professor at the School of Optics/CREOL at the University of Central Florida, in Orlando. She received an advanced engineering degree in optics from L'Ecole Superieure D'Optique in Orsay, France in 1984, and her Ph.D. from the Optical Sciences Center at the University of Arizona in 1990. Since 1984, she conducts research in optical system design, optical testing, and image understanding. As part of the later, she is currently investigating texture imaging and image analysis as they relate to medical, biomedical, and art. Her current research most generally includes development and assessment of technology for 3D visualization and imaging.