Abstract

This paper presents a range of techniques for texture segmentation using Kohonen self organizing feature maps (KSOFM). Various features, based on fractals and energy measures, are extracted from the texture images and then applied to the neural network. First, local features are extracted from the whole image using overlapping windows, then the feature vectors of each window are placed into the feature space. KSOFM is used as a clustering technique to assign a label to the groups that have similar features. A smoothing algorithm is performed followed by edge detection to give an edge map of the segmented image. An integrated software has been developed to illustrate the complete process of texture segmentation.

I. Introduction

Although there is no exact and generally accepted definition of texture, texture analysis is considered one of the most important subjects in image processing and computer vision. The task of extracting texture features is crucial and if one could model and quantify the process by which the human recognizes texture, one could construct a highly successful recognition system. Unfortunately, the process by which we recognize textures is not fully understood, and researchers are left to consider some ad hoc techniques. Conventional texture feature extraction algorithms can be grouped into four classes: statistical [1], structural [2], spectral [3], and model based [4, 5]. Statistical approaches yield characterizations of textures as smooth, coarse, grainy, and so on using relationship between intensity values of pixels; measures include the entropy, contrast, and correlation based on the gray level co-occurrence matrix. Structural algorithms are based on the image primitives; they regard primitive as the forming element to generate repeating patterns. Spectral techniques are based on properties of the Fourier spectrum and are used primarily to detect global periodicity in the image. Model based texture analysis methods are based on the construction of an image model that can be used not only to describe the texture, but also to synthesize it.

The various methods for modelling textures and extracting texture features can be applied in three broad categories of problems: texture segmentation, texture classification, and texture synthesis. Recently, many artificial neural network (NN) architectures for texture analysis have been proposed (for example, see [6, 7, 8]). Texture segmentation can be considered the most important problem since human can distinguish different textures quite easily, but the automatic segmentation is quite complex and it is still an open problem for research. Fractal geometry, proposed by Mandelbrot [9], was capable of describing textures images and some research has been done to estimate the fractal dimension of textures [10] but it has been found that different textures can have the same fractal dimension. So, multi-fractal features have been proposed in the literature [11] and proved to describe textures to some extent. Texture energy measure [12] is another feature that can be extracted from a texture image but like the fractal dimension, it can not characterize different textures. In this paper, a combination of fractal and energy features are used to segment textures using KSOFM unsupervised neural network to cluster the features into similar groups.
The sections of this paper are organized as follows. Section II reviews various features that can be extracted from a certain texture. These include: co-occurrence matrix features, energy measures, fractal measures, and multi fractal features. In Section III, the segmentation algorithm using Kohonen neural network is introduced. The developed software is given in Section IV. Section V presents the experimental evaluation of the proposed techniques. Finally, Section VI offers the conclusion.

II. Feature extraction

II.1 Co-occurrence matrix features

Spatial gray level co-occurrence estimates image properties related to second order statistics. In [1], the gray level co-occurrence matrices (GLCM) was suggested and became one of the most well-known texture features. The $G \times G$ GLCM, $P_d$, for a displacement vector $d = (dx, dy)$ is defined as follows. The entry $P_d(i,j)$ of $P_d$ is the pair of gray levels $i$ and $j$ which are a distance $d$ apart. It is given as

$$P_d = |\{(r, s), (t, v) : I(r, s) = i \land I(t, v) = j\}|$$

where $(r, s), (t, v) \in N \times N$, $(t, v) = (r + dx, s + dy)$, $G$ is the number of gray levels. The co-occurrence matrix reveals certain properties about the spatial distribution of the gray levels in the image. A number of useful texture could be obtained from the CLCM. Some of these features given next.

Energy = $\sum \sum P_d(i,j)P_d(i,j)$, Entropy = $-\sum \sum P_d(i,j)\log P_d(i,j)$, Contrast = $\sum \sum (i-j)^2P_d(i,j)$, and

Homogeneity = $\sum \sum \frac{P_d(i,j)}{1+|i-j|}$

These technique suffers from a number of difficulties such as the methods of selecting the displacement vector $d$ and finding the most relevant feature.

II.2 Texture energy measure

The texture energy measure, developed by Laws [12], will be used because its simplicity and effectiveness. These measures are derived from three basic one-dimensional templates, $L3 = [1, 2, 1]$ (dc level template), $E3 = [-1, 0, 1]$ (step template), and $S3 = [-1, 2, -1]$ (ripple template).

By convolution of these three basic templates, one can generate $1 \times 5, 3 \times 5, 5 \times 5, \ldots$ filter templates, each of them has its own interpretation.

$L5 = L3 \bullet L3 = \begin{bmatrix} 1 & 4 & 6 & 4 & 1 \end{bmatrix}$ (local average) \quad $S5 = L3 \bullet S3 = \begin{bmatrix} -1 & 0 & 2 & 0 & -1 \end{bmatrix}$ (spot detector)

$R5 = S3 \bullet S3 = \begin{bmatrix} 1 & -4 & 6 & -4 & 1 \end{bmatrix}$ (ripple detector) \quad $E5 = L3 \bullet E3 = \begin{bmatrix} -1 & -2 & 0 & 2 & 1 \end{bmatrix}$ (edge detector)

$W5 = -E3 \bullet S3 = \begin{bmatrix} -1 & 2 & 0 & -2 & 1 \end{bmatrix}$ (wave detector)

In order to use these masks to describe the textures in a subimage, we convolve them with the image and use the statistical characteristics of the result as a textural properties. Laws studied the power of these convolutions in conjunction with various statistics to discriminate textures. Based on his studies, he concluded that the most useful 4 masks are four zero sum obtained from $L5, S5, R5$, and $E5$. The most useful characteristics are either the sum of squares or the absolute value of each pixel of the image block after these masks are convolved with it. The four most useful $5 \times 5$ masks are given by:

$L5S5 = \begin{bmatrix} -1 & 0 & 2 & 0 & -1 \\ -4 & 0 & 8 & 0 & -4 \\ -6 & 0 & 12 & 0 & -6 \\ -4 & 0 & 8 & 0 & -4 \\ -1 & 0 & 2 & 0 & -1 \end{bmatrix}, \quad L5E5 = \begin{bmatrix} -1 & -2 & 0 & 2 & 1 \\ -4 & -8 & 0 & 8 & 4 \\ -6 & -12 & 0 & 12 & 6 \\ -4 & -8 & 0 & 8 & 4 \\ -1 & -2 & 0 & 2 & 1 \end{bmatrix}, \quad E5S5 = \begin{bmatrix} 1 & 0 & 2 & 0 & 1 \\ 2 & 0 & 4 & 0 & 2 \\ 0 & 0 & 0 & 0 & 0 \\ -2 & 0 & 4 & 0 & -2 \\ -1 & 0 & 2 & 0 & -1 \end{bmatrix}, \quad R5S5 = \begin{bmatrix} 1 & -4 & 6 & -4 & 1 \\ -4 & 16 & -24 & 16 & -4 \\ 6 & -24 & 36 & -24 & 6 \\ -4 & 16 & -24 & 16 & -4 \\ 1 & -4 & 6 & -4 & 1 \end{bmatrix}$
Assuming that $H(i,j)$ is one of these masks, $I_k(x,y)$ is the $k^{th}$ subimage, $O_k(x,y)$ is the $k^{th}$ corresponding output subimage. The texture energy of the $k^{th}$ window is $E_k$ and is given by

$$ E_k = \sum_{x=1}^{N} \sum_{y=1}^{N} O_k^2(x,y). $$

II.3 Fractal dimension

The fractal model introduced by Mandelbrot has been quite successful in modelling a wide variety of physical phenomena and natural objects such as clouds, trees, ..... A basic characteristic of fractals is the fractal dimension, which corresponds to the concept of roughness of the surface. The Hausdorff-Besicovitch (HB) or the fractal dimension of a bounded set $A$ is a real number used to characterize the geometrical complexity of $A$. A set is called a fractal set if its HB dimension is greater than its topological dimension. The texture features studied in this subsection are mainly based on the fractal geometry of the images. This choice is motivated by the observation that the fractal dimension, FD, is relatively insensitive to an image scaling and shows strong correlation with human judgement of the surface roughness.

The most common techniques for calculating the fractal dimension are the differential box counting and the triangular prism methods.

II.3.1 Differential Box Counting (DBC) [10]

A bounded set $A$ in the Euclidean $n$-space is self-similar if $A$ is the union of $N_r$ distinct (non-overlapping) copies of itself scaled up or down by a ratio $r$. The fractal dimension $D$ of $A$ is given by the relation:

$$ 1 = N_r r^D \quad \text{or} \quad D = \frac{\log(N_r)}{\log(1/r)} $$

The above equation is the basis for estimating the fractal dimension of an image by the DBC approach. $N_r$ in the above equation is determined in the following way. Let us assume that the image size is $M \times M$ pixels and it has been scaled down to a size $s \times s$ where $M/2 \geq s > 1$, $s$ is an integer, and $r = s/M$. Now considering the image as a 3D space with $(x,y)$ denoting 2D position and the third coordinate $(z)$ denoting gray level. The $(x,y)$ space is partitioned into grids of size $s \times s$. On each grid there is a column of boxes of size $s \times s \times (s/M) \times 256$. Let the minimum and the maximum gray level of the image in the $(i,j)^{th}$ grid fall in the $k^{th}$ and the $l^{th}$ box, respectively. Then, $N_r(i,j) = l - k + 1$ is the contribution of $N_r$ in the $(i,j)^{th}$ grid. Taking such contribution from all grids, we have $N_r$ counted for different values of $r$ (i.e. different values of $s$).

$$ N_r = \sum_{i} n_r(i,j) $$

We can estimate $D$, the fractal dimension, from the squares linear fit of $\log(N_r)$ against $\log(1/r)$.

II.3.2 Triangular prism surface area procedure (TPSA) [13]

The triangular prism surface area (TPSA) algorithm considers a gray scale image pixel P and a square environment with edges at the pixels A, B, C and D as shown in Fig. 1. The connections of the pixels’ gray scale values a, b, c, d, and p produce four triangles from which the area is taken. This is done for every image pixel that is far enough away from the image margin. A unique central pixel exists only in those cases where the size of the square $\tau$ is an odd pixel number. If $\tau$ is even, the mean of the gray scale value of the four pixels in the centre is taken to define the triangles as shown in Fig. 2.
The areas of all triangles for every central pixels are summed up to the entire area $s$ for different scales of $\tau$. The bilogarithmic plot of $s$ against $\tau$ should yield a linear line whose slope $\beta$ is used to determine the fractal dimension as $D = 2 - \beta$. The basic engine for this algorithm is similar to the box counting method, but slower due to the number of multiplications implied by the calculation of the areas.

II.4 Multi-fractals

Mandelbrot and Van Ness [14] have pointed out that different textures may have the same fractal dimension. This may be due to combined differences in the coarseness and directionality. So the idea of using a single fractal feature does not really come close to the solution for texture segmentation. Clearly, a single texture characterized by more than one fractal feature would probably result in a better segmentation. These different fractal features could be obtained by transforming the image in different ways and then taking the fractal dimension of these transformed images.

In this work we use, the features given in [11] are used. They are based on the FD of the original image and the transformed image as follows:
1-The original image.
2-The high gray level valued image
3-The low gray level valued image
4-The horizontal smoothed image
5-The vertical smoothed image.

III. Segmentation algorithms using KSOFM

For segmentation, we first calculate local features from the whole image, using overlapping windows, then place the feature vector of each window into the feature space, then using some clustering algorithms the windows with similar features are clustered into groups, and finally each group will be assigned a class label. Our algorithm will be based around self-organizing systems which can, as biological systems do, discover structure, patterns, or features directly from their environment. The Kohonen self organizing feature map (KSOFM) [15] is an unsupervised neural network model that strongly resembles in part the behaviour found in biological systems. Additionally, the SOFM, in part due to its biological roots possesses the unique property that cluster centres or features aggregate geometrically within the network output layer. That is, features that are similar by virtue of possessing a small Euclidean distance between them in feature space will simulate response in SOFM output neurons that are also geometrically close to each other.

The most common clustering technique is the K-means algorithm which is drawn from statistical pattern recognition [16]. We observe that the K-means method does possess self-organizing characteristics and we can notice similarities between this algorithm and the self organizing neural networks.
The Kohonen net architecture consists of two layers, an input layer and a Kohonen output layer. Each neuron in the input layer has a forward connection to each neuron in the output layer. The neurons of the output layer have lateral feedback connections. Suppose that an input pattern has \( N \) features and is represented by a vector \( X \) in an \( N \)-dimensional pattern space. The task that should be accomplished by the network is to map the input pattern to an output space. The output space is supposed to be one-dimensional or two-dimensional arrays of output nodes, which possess a certain topological order. In our texture segmentation problem, the input layer will have four neurons, since we have four features, and the output layer will have the number of clusters which initially must be provided. The 1D KSOFM is shown in Fig. 3.

![Fig. 3. Lateral feedback in 1D KSOFM neural network](image)

The training phase of the network is called competitive learning or winner take all technique and it can be illustrated in the following steps:

**a-** First a winning neuron is selected as the one with the shortest Euclidean distance \( \| x - w_i \| \) between its weight vector and the input vector, where \( w_i \) denotes the weight vector corresponding to the \( i^{th} \) output neuron.

**b-** Let \( i^* \) denotes the index of the winner neuron and let \( I^* \) denotes a set of indexes corresponding to the neighbourhood of winner \( i^* \). Then the weights associated with the winner and its neighbouring neurons are updated by \( \Delta w_j = \eta(x - w_j) \) for all the indices \( j \in I^* \), and \( \eta \) is a small positive learning rate. The amount of updating may be weighted according to a preassigned neighbourhood function, \( \Lambda(j, i^*) \) for all \( j \). For example, a neighbourhood function may be chosen as

\[
\Lambda(j, i^*) = \exp \left( -\frac{(r_j - r_i)^2}{2\sigma^2} \right)
\]

This function represents the position of the neuron \( j \) in the output space. The convergence of the feature map depends on a proper choice of \( r_i \). The size of the neighborhood, \( \sigma \) should decrease gradually.

**c-** The weight update should be immediately succeeded by the normalization of \( w_i \). In the retrieving phase, all the output neurons calculate the Euclidean distance between the weights and the input vector and the winning neuron is the one with the shortest distance.

### IV. Developed software

A C++ software package has been developed to illustrate the various segmentation techniques proposed in the paper. The following steps give a brief description of the functions that are performed by the package.

**A. Reading the input texture image**

The user provides the name of a texture bitmap file (BMP format) and the program opens the file and gets the grey level values of the read image and store them in a matrix. The user also provides the size of the processing window, \( R \), and the separation step between windows as shown in Fig. 4. Typically the size of the window is 32 or 16 and the step is 1 or 2.
B. Feature calculation:
The second set of parameters that should be entered is the feature type that will be calculated. There are two selections, either calculating the features using the fractal dimension and texture energy or the cooccurrence matrix features. If the user chooses the first one, he will be asked for the methods of calculating the fractal dimension, as illustrated in Section II.3 (differential box counting method or triangular prism surface area method). After that, the technique of calculating the multi fractal features should be provided as illustrated in Section II.4 (high and low gray level or horizontal and vertical smoothing techniques). The mask given in Section II.2 is used for calculating the texture energy. On the other hand, if the user selects the cooccurrence matrix features, the four features (contrast, energy, entropy, and homogeneity) given in Section II.1 will be calculated for displacement \( d = 1 \) and all directions (0, 45, 90, and 135) and then taking the average of each feature.

C. Segmentation implementation
The user will be asked to choose one of two segmentation techniques, K means clustering or Kohonen self organizing map. The user should also provide the number of clusters into which the image will be segmented. Since the number of clusters is not known in advance, it is a good practice to provide a relatively large number of clusters and let the algorithms iterate until reaching the exact number of clusters in the image. The input to the segmentation algorithm will be the features matrix shown in the Fig. 5. The output will be the features matrix but having one more column containing the cluster number of each feature vector.

D. Constructing the output image
This is accomplished by giving each pixel a gray level proportional to the cluster number, so each of the textures constituting the image will have the same gray level in the output image. To enhance the output image and sharpen the boundaries between different clusters, the program asks the user to choose a filtering technique (mean, median, Gaussian, or Kuwahara filters) and edge detection technique (Sobel or Roberts edge detectors). The final segmented image will be saved as a bitmap file (output.bmp) and can be displayed by any graphics program. After scanning the output image, if the user wants to rerun the program for different number of clusters or wants to apply different segmentation technique, this can be done without recalculating the features so the processing will be much faster.

Fig. 4 Input image and overlapping windows
Fig. 5 Feature matrix
V. Simulation results

In this section, we present some examples of texture segmentation using the techniques discussed in the paper. The simulation has been conducted on synthetic textures and natural textures.

VI.1 Segmentation of synthetic texture images

As shown in Fig. 6a, a 256 x 256 synthetic texture that has four different textures has been prepared using a graphics program. Fig. 6b shows the segmentation using KSOFM with 5 clusters. A Roberts edge detection algorithm has been applied and the result is shown in Fig. 6c.

![Fig. 6. Segmentation of synthetic textures a) original synthetic texture, b) segmented image using KSOFM with 5 output neurons, and c) segmented image after applying Roberts edge detection algorithm.](image)

It is obviously clear that the KSOFM was able to segment the texture image into exactly four different textures. It assigns the gray levels 60, 90, 120, and 150 to the four regions in Fig. 6b and 6c. These successful segmentation results from the fact that the used synthetic textures were deterministic, i.e., each of them can be described by the characteristics of one subpattern or primitive. This is not the same case for natural textures.

VI.2 Segmentation of natural texture images

To conduct this simulation, different images were constructed using Brodatz album textures [17]. First, we used a two 400 x 400 pixel mosaic images, shown in Fig. 7, with nine different textures in each. The developed software calculates the features as illustrated in the previous section and apply the KSOFM, for segmentation. The natural textures case is not as simple as the synthetic textures because the natural textures are stochastic and rely on the spatial distribution of the primitives.

The parameters on which the simulation has been carried out are:
- Window size: 32 pixels
- Step size: 2 pixels.
- Energy mask: 5 x 5 Laws mask
- Fractal dimension calculation method: differential box counting.
- Multi fractal calculation method: low and high gray level technique.
- Number of clusters: 20.
- Segmentation technique: KSOFM.

It is apparent that the proposed algorithm could successfully segment the image into distinguishable classes.
It was also noticed that the KSOFM technique outperforms the conventional K-means clustering technique in the speed and the quality of the segmentation.

**VII. Conclusion**

In this paper, the power of self-organization has been demonstrated on a hard real world problem, the segmentation of stochastic textures. A new and practical method using Kohonen self organizing map neural network has been developed. First different features are extracted from the image using fractal and other statistical techniques and then these features are applied to the neural network, which works as an unsupervised clustering technique. Simulation has been carried out on synthetic as well as natural textures. In both cases, the algorithm was able to correctly segment the texture images into different groups.
References


