Image and Vision Computing 15 (1997) 937–948

An MLP-based texture segmentation method without selecting a feature set

U. Bhattacharya, B.B. Chaudhuri*, S.K. Parui

Computer Vision and Pattern Recognition Unit, Indian Statistical Institute, 203, Barrackpore Trunck Road, Calcutta, 700 035 India

Received 9 September 1996; revised 2 April 1997; accepted 8 April 1997

Abstract

A texture segmentation technique which employs a multilayer perceptron (MLP) and does not consider the selection of features is presented in this paper. Thus, users can avoid selection and computation of the feature set and hence real-time segmentation may be possible. The technique apparently works in a fashion similar to our visual system whereby we do not consciously compute any feature for texture discrimination. A detailed study has been made for the selection of the network size. A newly proposed variant of the back-propagation algorithm has been used for more efficient training of the network. An edge-preserving noise-smoothing approach has been proposed to remove noise from the segmented image.

Keywords: Texture segmentation; Multilayer perceptron; Modified backpropagation; Feature extraction

1. Introduction

Texture plays an important role in image processing and computer vision problems because most objects encountered in real life have textured surfaces. An image may contain regions of different textures where it may be necessary to segment the region according to the textural signature. This process, known as texture segmentation, is useful in various applications such as remote sensing, surface inspection, geology, biomedical imaging, etc.

Texture segmentation is different from conventional object and region segmentation because of the large variation of gray value and existence of edges over a region of uniform texture. Thus, texture homogeneity is different from gray level homogeneity and the boundary between two different textured regions is not a conventional edge. A wide variety of texture segmentation approaches is available in the literature (Haralick [1], Gool et al. [2], Tuceryan and Jain [3] and Reed and du Buf [4] made comprehensive surveys on texture analysis techniques). These segmentation approaches can be loosely classified as feature-based, model-based and structural methods [4]. The first category involves various statistical moments [5], co-occurrence matrix based features [6], fractal features [7], etc. Texture classification on the basis of these features is usually expensive in terms of computation. The second category is based on autoregressive models [8], Markov random field models [9] etc., and the last category is based on specific primitives and certain placement rules governing their spatial interactions [4].

Recently, there has been considerable interest in using artificial neural networks (ANN) for solving computer vision problems—the main reasons being the parallel architecture of a connectionist network model and its learning capability in an adaptive manner. Moreover, an ANN-based segmentation approach has some biological plausibility because the computational models of the brain are largely characterized by highly interconnected information processing units.

The ANN model used for texture segmentation in the supervised framework is normally the multilayer perceptron (MLP) network model. Shang and Brown [10] described a texture classification system using MLP networks—they used features extracted by some conventional way such as the co-occurrence matrix method and then employed principal component transformation to reduce the number of features. Farrokhnia [11] used a single hidden layer network and backpropagation (BP) algorithm with Gabor transform features for texture segmentation. Haddon and Boyce [12] have used MLP neural networks together with the BP learning algorithm for texture segmentation where a subset of Hermite feature space is presented to the network as the input feature vector. An MLP network-based two-stage multiresolution approach to texture segmentation was proposed by Yhann and Young [13] using line and edge features.

One common aspect of the methods described above is the presence of a set of predefined features (or a set of primitives along with a set of placement rules) that should be selected and computed before segmentation. But the human visual system does not seem to distinguish textures by consciously using a predetermined feature set. In childhood a human being learns to classify different objects just by examples and the determination of the feature set in the visual cortex, if any, is not transparent to him.

In the present paper, we describe an MLP-based texture segmentation method that tries to act like a human observer, in that it is not necessary to select and compute a set of features before classification. Thus, the image is presented to the input layer of the MLP network and we obtain a texture segmentation mapping at the output, without bothering about what features are extracted in the intermediate layers. Our study is to the best of our knowledge distinct from all other studies of MLP-based texture segmentation in this respect. Another contribution in this paper is a refinement of the original BP algorithm for better convergence performance of the training of an MLP network. Using the original BP algorithm the training process quite often either oscillates or converges extremely slowly. The robustness of the proposed method, described in the next section, will be clear from the simulation results.

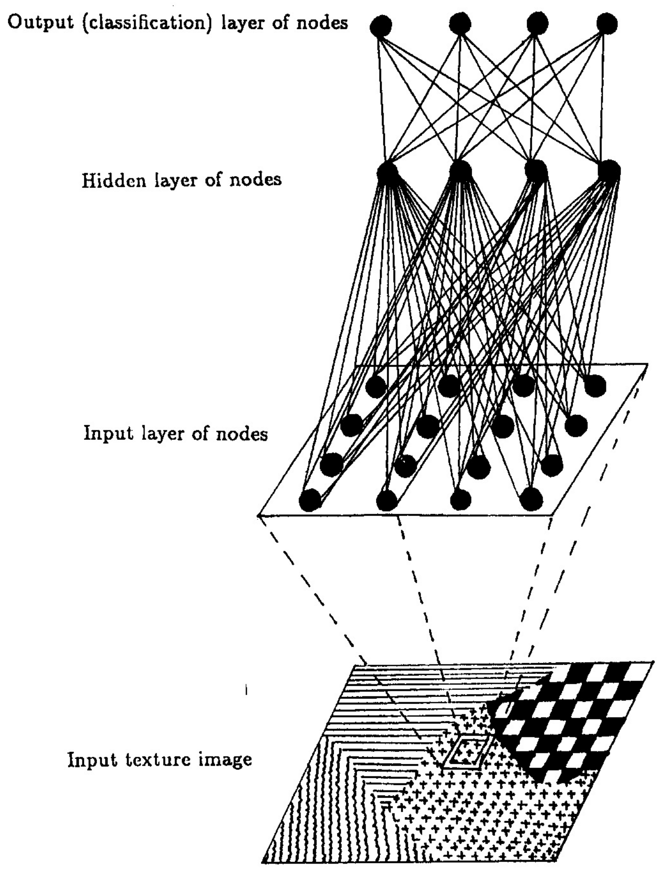


Fig. 1. An MLP-based texture classifier.

2. Texture segmentation using multilayer perceptrons

In the proposed approach to texture segmentation the pixel gray values within certain windows are presented to the single hidden layer MLP network. The network itself calculates the distinguishing feature values in its hidden nodes that are not transparent to the user.

The structure of the proposed system is shown in Fig. 1. Here a small window of $n \times n$ pixels is presented to the network at a time. Thus, the input layer of the MLP network has $n \times n$ nodes. These nodes are fully connected to m hidden layer nodes which are in turn fully connected to the k output nodes, where k is the number of texture classes under consideration. The network is initially trained using a training set of patterns (or windows). Once trained, it is used for classifying unknown scenes.

Suppose a window U is presented to the network for classification. If the qth output node has maximum value, then the central pixel of U is assigned to the qth texture class. The process is repeated for windows at all positions of the given textured image and a classification mapping is thus obtained. The mapping sometimes contains small isolated regions of misclassified pixels, which is called classification noise. Hence the mapping is smoothed and artifacts are removed to get the final segmentation mapping.

The MLP network is trained using a modified backpropagation algorithm. We have studied the selection of network size and learning parameter values in details. For removal of classification noise in the segmented image, we have considered the edge-preserving noise-smoothing octant (EPNSO) technique, which is very similar to the well-known edge-preserving noise-smoothing quadrant (EPNSQ) filtering approach. Moreover, we have used a component labelling based approach to remove the false boundaries.

2.1. Selection of training samples

A training sample consists of a window of size $n \times n$, and for a proper choice of training samples, the following points are noted:

1. texture primitives may be randomly or regularly arranged;
2. there may be uneven and slow tonal variation over the texture field;
3. there seems to be a minimum area over which the texture should be viewed for recognition and this area seems to be different for different textures.

These factors make the choice of representative samples an extremely difficult task. Keeping them in mind, we have selected 150 training examples from each texture class in the following way. From the photograph of each texture class (from the Brodatz album [14]), we have manually selected three 100×100 sub-images with maximum, moderate and minimum homogeneity respectively. From each of these sub-images, 50 pixels have been selected randomly using uniform distribution. Around each of these randomly selected pixels, a window of size $n \times n$ is chosen as a training sample. Each training sample is normalized with respect to the maximum possible gray value 255.

2.2. *The backpropagation algorithm and its modifications*

The training of an MLP network is usually done using BP algorithm [15]. This algorithm performs a gradient descent, corresponding to each pattern p in the training set, in the connection weight space on an error surface defined by

$$E_p = \frac{1}{2}\sum_k (t_{pk} - y_{pk})^2 \tag{1}$$

where $\{t_{pk}\}, \{y_{pk}\}$ are, respectively, the target and output vectors corresponding to the pth input pattern. The system error E is defined as

$$E = \frac{1}{P}\sum_p E_p \tag{2}$$

where P is the total number of patterns in the training set. In the BP algorithm, weight modification rules are given by

$$w_{jk}(t+1) = w_{jk}(t) - \eta \frac{\partial E_p(t)}{\partial w_{jk}(t)} \tag{3}$$

$$w_{ij}(t+1) = w_{ij}(t) - \eta \frac{\partial E_p(t)}{\partial w_{ij}(t)} \tag{4}$$

where $w_{jk}(t)$ is the weight connecting a hidden node j with an output node k while $w_{ij}(t)$ is the weight connecting an input node i with a hidden node j at time t and η is a positive constant, called the learning rate.

As the BP algorithm performs a gradient descent on a hyper surface, called error surface, in the weight space, there is no guarantee that the global minimum of that surface will be reached (even approximately) after a moderate number of sweeps, and the algorithm may get stuck at a local minimum. The difficulty in reaching the global minimum on the error surface from a random initial position is dependent on the nature of the surface. To tackle the problem Rumelhart et al. [15] suggested modification of the above weight modification rule by including a momentum term.

Though this suggestion is effective in solving different benchmark problems, we have observed that a non-zero momentum term in the weight modification rule is counterproductive in the texture segmentation problem. Below we present a modification of the BP algorithm in order to achieve faster and more robust convergence.

2.2.1. *Self-adaptation of learning rates*

The selection of a suitable step size (learning rate) is an issue common to all steepest-descent methods. A large value of the learning rate often leads to oscillation whereas a small value causes very slow convergence to the desired minimum on the error surface. Bhattacharya and Parui [16] observed that the single layer perceptron learning algorithm can show improved performance if the learning rate is varied over time under some constraints. This observation is also valid for the multilayer perceptron [17]. During our extensive study on the convergence of the BP algorithm we have made the following inferences, the first four of which were first made by Jacobs [17].

1. Every weight of the network should have its own individual learning rate.
2. Every learning rate should be allowed to vary over time.
3. When the error derivative with respect to a weight possesses the same sign for several consecutive steps, the learning rate for that weight should be increased.
4. When the sign of the derivative with respect to a weight alternates for several consecutive steps, the learning rate for that weight should be decreased.
5. The modifications to the learning rates at any time should be entirely based on the shape of error surface at the present position.
6. The overall convergence performance of the algorithm should not depend much on the choice of the parameter values involved in the modification rule for the learning rates.
7. The values of the learning rates should not be allowed to increase indefinitely so that the weight values do not explode.

A newly suggested method for the self-adaptation of learning rates consists of performing a gradient-descent on the surface defined by Eq. (1). Using the self-adaptive learning rates, the weight modification rules given by Eqs. (3) and (4) become

$$w_{jk}(t+1) = w_{jk}(t) - \beta_{jk} \frac{\partial E_p(t)}{\partial w_{jk}(t)} \tag{5}$$

$$w_{ij}(t+1) = w_{ij}(t) - \beta_{ij} \frac{\partial E_p(t)}{\partial w_{ij}(t)} \tag{6}$$

where $\beta_{jk} = h(\eta_{jk})$ and $\beta_{ij} = h(\eta_{ij})$; $h(x) = d/(1 + e^{-x + d/2})$ is called the effective value function and $d > 0$ is a constant. This effective value function has been considered to avoid the danger of self-adaptive learning rates growing very large [18].

The modification rules for learning rates are:

$$\eta_{jk}(t+1) = \eta_{jk}(t) + \Delta_p \eta_{jk}(t) \tag{7}$$

and

$$\eta_{ij}(t+1) = \eta_{ij}(t) + \Delta_p \eta_{ij}(t) \tag{8}$$

where

$$\Delta_p \eta_{jk}(t) = \frac{\gamma}{d} \frac{\partial E_p(t)}{\partial w_{jk}(t)} \frac{\partial E_p(t-1)}{\partial w_{jk}(t-1)} \beta_{jk}(d - \beta_{jk})$$

and

$$\Delta_p \eta_{ij}(t) = \frac{\gamma}{d} \frac{\partial E_p(t)}{\partial w_{ij}(t)} \frac{\partial E_p(t-1)}{\partial w_{ij}(t-1)} \beta_{ij}(d - \beta_{ij})$$

where γ is a constant of proportionality.

It has been observed that the learning performance of an MLP network using this modified BP algorithm does not depend much on the choice of γ. In the present study, the value of γ is always taken as 0.1. The constant d determines the maximum value that can be assumed by a learning rate. In the present problem of texture segmentation, the proper value of η has always been found to be less than 1.0 and values more than 1.0 almost always lead to oscillation. Consequently, the value of d is always taken to be 1.0. In problems where a larger range for η is needed, d is assigned a larger value.

2.2.2. *Termination of the learning session*

Overlearning is a common problem of the BP algorithm and setting up of an appropriate condition for the termination of the learning session is very important. To avoid overlearning, we have considered a different set of patterns, called the test set, and fed it also to the network during learning. The test set has been constructed independently in a manner similar to the training set described earlier.

The test set has been used to determine the termination point of the learning session. During the learning session, the average system error is computed after each sweep, for both the training and test sets. Initially, it is found that this error on both the training and test sets is decreasing. But after some number of sweeps, the error on the test set goes on increasing though the error on the training set still decreases. The point of time when the error on the test set increases for at least 3 consecutive sweeps for the first instance is noted and the weight values before the error started increasing is stored. This behavior of the learning of the connection weights in an MLP is shown in Fig. 2.

2.3. *Network architecture selection*

As mentioned before, we used one hidden layer in the MLP network for texture segmentation. The reason for using only one hidden layer is that we have observed that the use of multiple hidden layers deteriorates the training performance in terms of rate of convergence. As described below we have studied in detail the selection of the number of nodes in the input and hidden layers of the network. The input pattern to the network consists of the pixel gray values within a square window and hence the number of input nodes is specified by the size of such a window. The number of output nodes is the number of texture classes considered.

2.3.1. *Selection of window size*

Using input windows of smaller size can reduce the computational burden. But in many situations small-sized windows may not capture the properties of a texture. A large window size contains sufficient texture information, but demands huge computation. The choice of optimal window size is an important aspect of the proposed segmentation method. In fact, we have studied the effect of various window sizes on the performance of the system using different texture mosaics from the Brodatz album [14]. The window sizes considered are 3×3, 7×7, 11×11 and 15×15. In most cases, the 15×15 window size provided good training performance with respect to the rate of convergence. But the approximation of real boundaries in the texture mosaic is not very satisfactory for such a large window. Moreover, for such a large-sized input layer the computational load also increases. Though the optimum window size depends on the nature of the textures under consideration, we have observed that in most of the cases considered by us, 11×11 windows provide minimum segmentation error among all four different choices of window size and the number of sweeps required for the training is also moderate (Table 1).

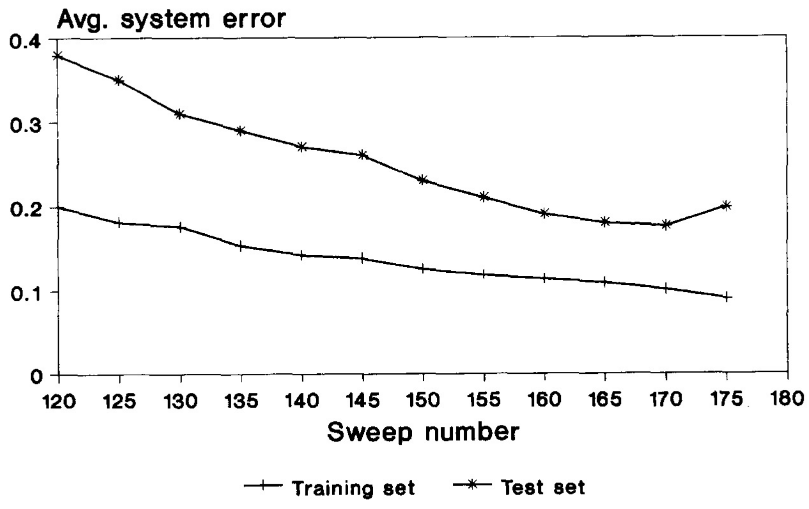


Fig. 2. The behavior of the training of an MLP on training and test sets of patterns (here the segmentation task of Fig. 7(a) is considered with optimal parameter values).

Table 1
Input window size effect on the performance of the proposed segmentation technique (when texture mosaic in Fig. 7(a) is used)

Window size	Segmentation mapping	Average mean square error (%) on the training set	Average classification accuracy (%) on the test image	Average no. of sweeps required
3×3	7(b_1)	0.27	73.18	385
7×7	7(b_2)	0.19	95.10	305
11×11	7(b_3)	0.09	95.93	175
15×15	7(b_4)	0.07	94.0	85

2.3.2. Selection of the number of hidden nodes

The capacity of a multilayer perceptron to approximate a given mapping has been investigated by Hornik et al. [19]. But in almost all of the real-life applications of such a network, the minimum number of hidden units required to accomplish the underlying mapping cannot be obtained in a straightforward manner, and the selection of this number involves conflicting interests [20]. After conducting experiments on several texture mosaics of four textures with different hidden layer sizes, we found that for an 11×11 input layer a hidden layer of size 12 is optimum in terms of learning and classification performances (Table 2). We have noted that mosaics of 2 and 6 textures can be effectively tackled using 4 and 20 nodes in the hidden layer. However, for larger number of textures in a scene the number of nodes should be increased. A rule of thumb about the minimum number of nodes in the hidden layer may be $4(r - 1)$ where r is the number of different textures in the scene.

3. Segmented image space smoothing

The proposed texture segmentation technique may result in considerable misclassification in some cases and the misclassified pixels usually occur in the form of speckles scattered here and there. A segmented image space smoothing can reduce the misclassification to a large extent and an edge-preserving smoothing technique such as in [21,22] appears suitable.

Jiang and Sawchuk [22] suggested the EPNSQ filtering method. In this method quadrant windows around each pixel are used to estimate local statistics and the average value of the gray levels of the pixels belonging to the most homogeneous window is used to replace the pixel under consideration.

The proposed smoothing technique is similar to the EPNSQ filtering method but, unlike the EPNSQ method, it is suitable for application on a segmented image. To implement the algorithm, the segmentation mapping is given distinct class labels. If the scene contains k texture classes, then a pixel may be assigned a class label from $\{1,2,\ldots,k\}$. Now, the segmentation mapping may be considered as an image in which the gray value of a pixel is its class label. Let $f(i,j)$ denote the gray value (class label) of the pixel having coordinates (i,j) in the segmentation mapping image. We shall smooth such an image using the following approach.

Consider all eight octants within a $(2U + 1) \times (2U + 1)$ neighborhood around the candidate pixel at (i,j) (see Fig. 3). The variances on each of the octants are computed as follows:

$$V_{U_1}(i,j) = \frac{1}{N}\sum_{l=0}^{U}\sum_{m=l}^{U}[f(i+l;j+m)]^2 - \left[\frac{1}{N}\sum_{l=0}^{U}\sum_{m=l}^{U} f(i+l,j+m)\right]^2$$

$$V_{U_2}(i,j) = \frac{1}{N}\sum_{l=0}^{U}\sum_{m=0}^{l}[f(i+l,j+m)]^2 - \left[\frac{1}{N}\sum_{l=0}^{U}\sum_{m=0}^{l} f(i+l,j+m)\right]^2$$

Table 2
Hidden layer size effect on the performance of the proposed segmentation technique (when texture mosaic in Fig. 7(a) is used)

Hidden layer size	Segmentation mapping	Average mean square error (%) on the training set	Average classification accuracy (%) on the test image	Average no. of sweeps required
4	7(b_5)	0.33	87.55	527
8	7(b_6)	0.21	95.26	211
12	7(b_7)	0.09	95.93	175
16	7(b_8)	0.08	95.50	164
20	7(b_9)	0.08	95.14	112

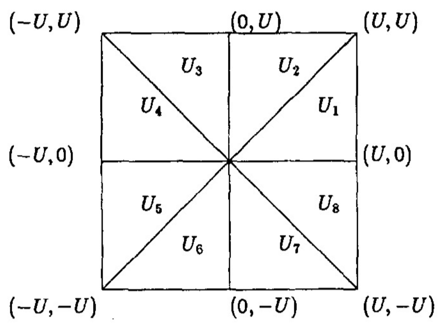


Fig. 3. Schematic representation of the octants of the EPNSO technique.

$$V_{U_3}(i,j)=\frac{1}{N}\sum_{l=0}^{U}\sum_{m=0}^{l}[f(i+l,j-m)]^2-\left[\frac{1}{N}\sum_{l=0}^{U}\sum_{m=0}^{l}f(i+l,j-m)\right]^2$$

$$V_{U_4}(i,j)=\frac{1}{N}\sum_{l=0}^{U}\sum_{m=l}^{U}[f(i+l,j-m)]^2-\left[\frac{1}{N}\sum_{l=0}^{U}\sum_{m=l}^{U}f(i+l,j-m)\right]^2$$

$$V_{U_5}(i,j)=\frac{1}{N}\sum_{l=0}^{U}\sum_{m=l}^{U}[f(i-l,j-m)]^2-\left[\frac{1}{N}\sum_{l=0}^{U}\sum_{m=l}^{U}f(i-l,j-m)\right]^2$$

$$V_{U_6}(i,j)=\frac{1}{N}\sum_{l=0}^{U}\sum_{m=0}^{l}[f(i-l,j-m)]^2-\left[\frac{1}{N}\sum_{l=0}^{U}\sum_{m=0}^{l}f(i-l,j-m)\right]^2$$

$$V_{U_7}(i,j)=\frac{1}{N}\sum_{l=0}^{U}\sum_{m=0}^{l}[f(i-l,j+m)]^2-\left[\frac{1}{N}\sum_{l=0}^{U}\sum_{m=0}^{l}f(i-l,j+m)\right]^2$$

$$V_{U_8}(i,j)=\frac{1}{N}\sum_{l=0}^{U}\sum_{m=l}^{U}[f(i-l,j+m)]^2-\left[\frac{1}{N}\sum_{l=0}^{U}\sum_{m=l}^{U}f(i-l,j+m)\right]^2$$

where $N=(1/2)(U+1)(U+2)$ and $V_{U_n}(i,j)$, $n=1,2,\ldots,8$, represent the variance computed on the octant U_n around the (i,j)th pixel.

If $V_{U_q}(i,j)$ is the minimum among all the $V_{U_n}(i,j)$, $n=1,2,\ldots,8$, then according to the present method, called the edge-preserving noise-smoothing octant (EPNSO) filtering approach, the modal value of the class labels of the pixels in U_q is found and the gray value of the (i,j)th pixel is replaced by this modal gray value. We have observed that the EPNSO method performs better than the EPNSQ in preserving edges. Moreover, this smoothing technique has been proposed specifically to be applied on the segmented image, not on the original image.

This smoothing technique with a certain fixed window size can be applied on the segmented image repeatedly until no further improvement is observed in two successive applications of the technique. This method is very effective at reducing the noise in the segmented image by keeping the real edges unaffected. But sometimes it may not be able to remove patches of small area. To remove them, at first such small patches are identified by component labelling and the label of the pixels in such a patch is changed to the label of the pixels maximally surrounding the patch. The EPNSO method followed by component labelling provides very good segmentation results. This post-processing method has resulted in improvement of correct classification of textures by about 4% on average in our study.

4. Simulation results

We have simulated the proposed MLP-based texture segmentation technique on a set of texture-mosaic images taken from the Brodatz album [14]. As per our experience a region of reasonable size usually consists of at most four different textures in its surroundings. It is clear that with the increase in the number of texture classes, the complexity of learning tasks increases. To obtain the simulation results described below, we have trained the respective MLP networks using the BP algorithm with self-adaptive learning rates. The mosaics considered in the simulation consist of test images selected from the respective texture classes [14]. These test images are different from the training and test data (which have been used for learning of the respective MLP networks). The statistics on the classification performance reported later are based on such additional test images.

In each of the segmentation tasks considered in the present paper, the respective networks have been separately trained. It is to be noted that a suitably chosen network architecture can learn a good number of texture classes although the complexity of the learning task increases with a larger number of different classes.

In Fig. 4(a_1–a_8) eight mosaics each consisting of two different textures are shown. Each mosaic is of size 128 × 128 pixels so that the textured image from each class is of size 128 × 64 pixels. The images are quantized in 256 gray levels. In Table 3, we present the learning statistics corresponding to each of these eight two-class segmentation tasks. The input window size and the number

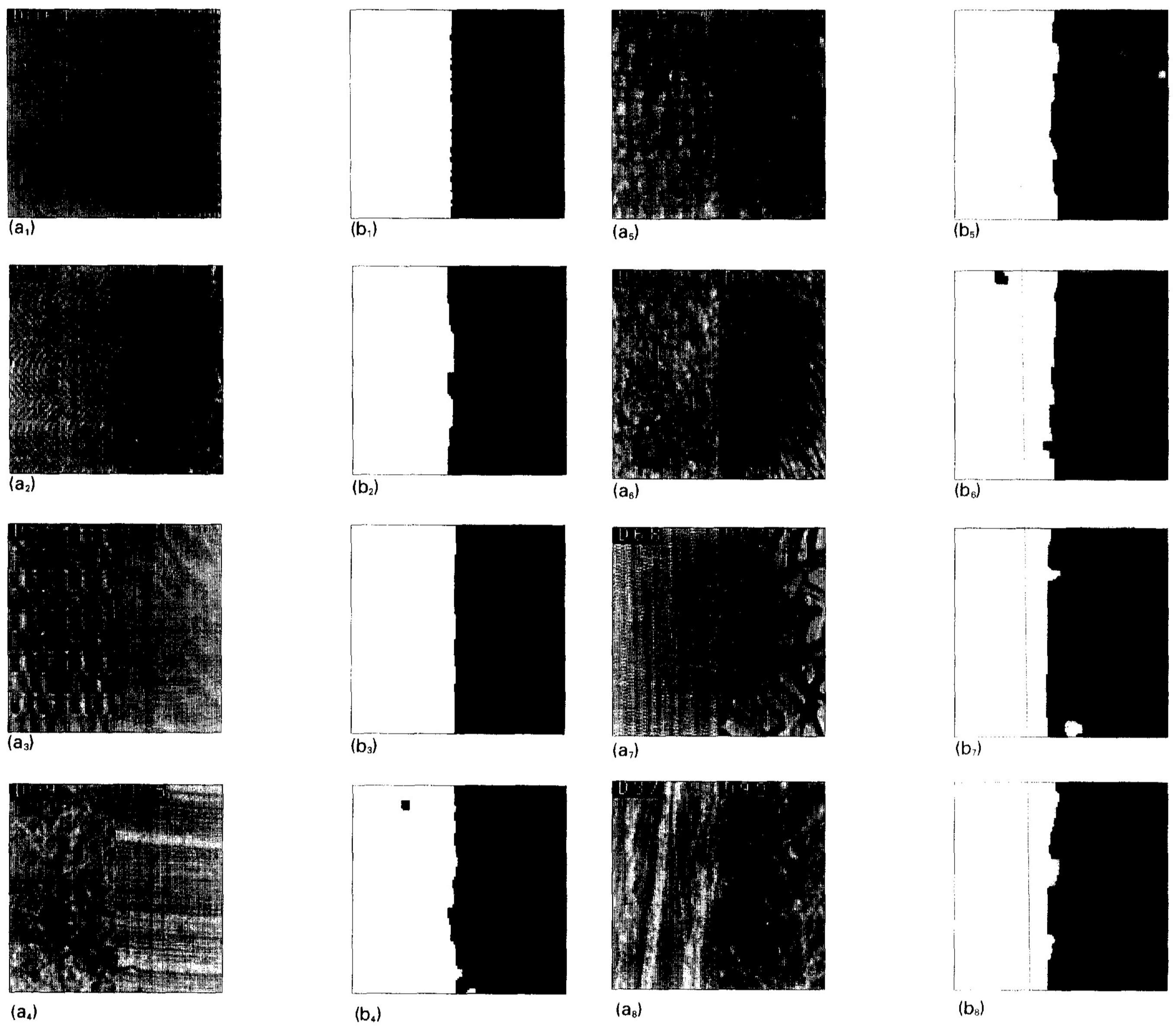


Fig. 4. (a_1–a_8) Different mosaics of natural textures taking two at a time; (b_1–b_8) the corresponding segmentation mappings.

of hidden nodes in this table correspond to approximately the best selections of the minimal network sizes in respective cases. The corresponding results of segmentation are shown in Fig. 4(b_1–b_8), respectively. In most cases the segmented image space smoothing technique EPNSO is performed on windows of size 9 × 9 two to three times.

It is noted that misclassifications occur mainly at the border of two adjoining texture classes. However, tolerance to boundary curvature is depicted in Fig. 5 while correct

Table 3
Learning and network statistics corresponding to segmentation tasks for mosaics in Fig. 4

Texture mosaic	Input window size	No. of hidden nodes	No. of sweeps	Initial value of η	Final system error	% of correct classification
4(a_1)	7 × 7	4	52	0.1	.015	98.72
4(a_2)	7 × 7	4	76	0.1	.019	97.84
4(a_3)	7 × 7	4	96	0.1	.018	99.30
4(a_4)	7 × 7	4	188	0.1	.033	98.34
4(a_5)	7 × 7	4	146	0.1	.039	97.95
4(a_6)	7 × 7	8	115	0.1	.023	97.87
4(a_7)	11 × 11	8	128	0.1	.057	96.59
4(a_8)	11 × 11	8	165	0.1	.036	97.98

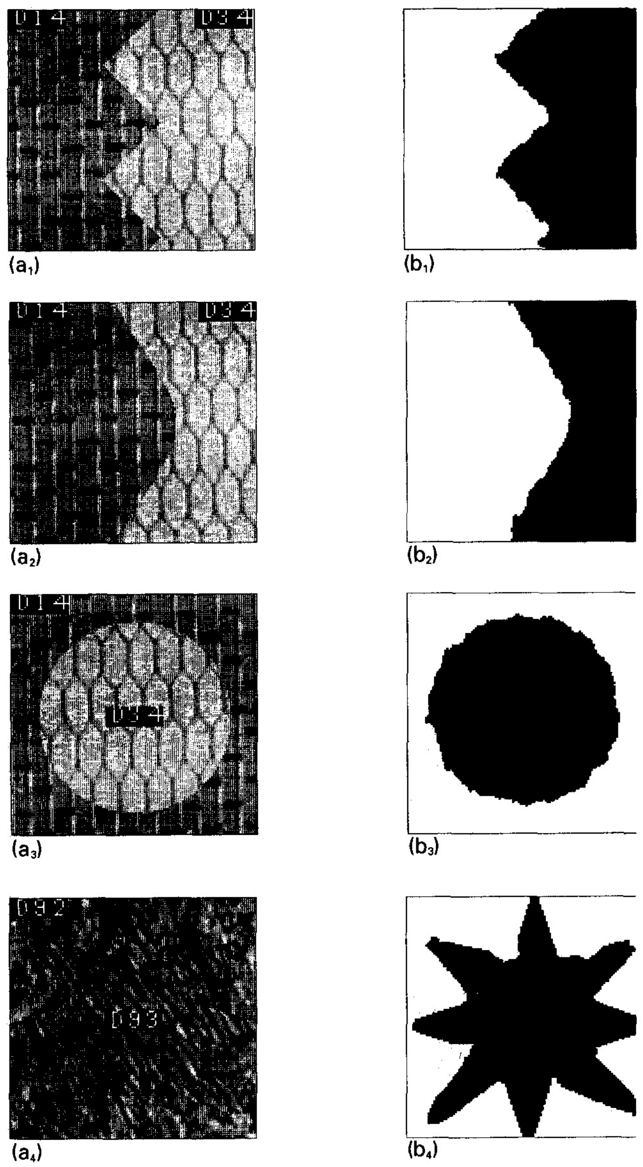


Fig. 5. (a_1–a_4) Four mosaics of natural textures depicting different boundary shapes; (b_1–b_4) the corresponding segmentation mappings.

classification percentages are shown in Table 4. In these cases too the EPNSO technique has been performed two to three times.

In Fig. 6(a_1–a_5) five mosaics consisting of four, five, six, nine and twelve different textures, respectively, are shown. The learning statistics and percentages of correct classification in these segmentation tasks are presented in Table 5. The corresponding final segmentation mappings are shown in Fig. 6(b_1–b_5). In each of these cases the EPNSO (using windows of size 9×9) smoothing technique is applied three to four times.

For the purpose of demonstrating the effect of input window size and the number of hidden nodes we have chosen the texture mosaic in Fig. 7(a). The segmentation mapping using different window sizes are shown in Fig. 7(b_1–b_4). In these simulations 12 hidden nodes have been used. Various results related to this study are summarized in Table 1.

In this table, the averages are taken over ten different random initializations of connection weights. It is seen from this table that a window of 11×11 pixels can be accepted as the optimal window size because in this case the minimum average segmentation error could be achieved after moderate average sweep numbers. Also, for this window size the average mean square error on the training set of data is moderate. In fact, in most of the mosaics we have experimented with, 11×11 is found to be the optimal window size with respect to the average segmentation error on the test image among the four choices we considered.

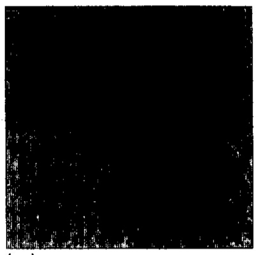
Also, the segmentation mappings of the above set of mosaics using different number of hidden nodes are shown in Fig. 7(b_5–b_9). In all of these segmentations, 11×11 input windows have been used. The results obtained are given in Table 2.

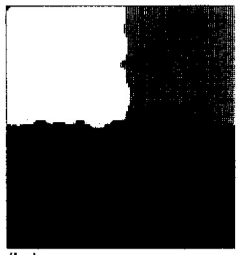
In most 4-class texture segmentation problems, it has been observed that 12 nodes in the hidden layer give the optimal choice. For 2-class and 6-class problems, 4 and 20 hidden nodes, respectively, are found (assuming 11×11 input windows) to be good choices.

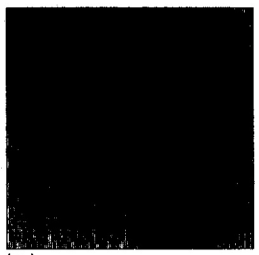
To compare the performance of the proposed technique with one of the existing methods of texture segmentation, we have considered the technique using four fractal geometry based features [23]. In Table 6 we use the classification accuracy statistics given in [23] and compare them with those obtained using our method. The texture mosaics used for this study are shown in Fig. 8(a_1–a_4) and the corresponding segmentation mappings obtained using the proposed approach are shown in Fig. 8(b_1–b_4) (the results are obtained after applying the EPNSO smoothing followed by the component labelling technique for removing noise). Fifty training samples were used for computing the centroid for the minimum distance (Min-dist) classifier. In the k-nearest neighbor (kNN) technique, the chosen values of k were 1 and 5 while the numbers of training samples were 25 and 50 (denoted by A and B respectively). From this table it is seen that the performance of the proposed MLP-based segmentation technique is comparable with

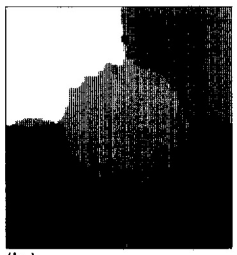
Table 4
Percentage of correct classification for mosaics in Fig. 5

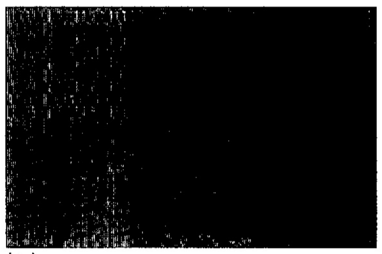
	Texture mosaic			
	5(a_1)	5(a_2)	5(a_3)	5(a_4)
% of correct classification	97.52	97.18	92.40	94.98

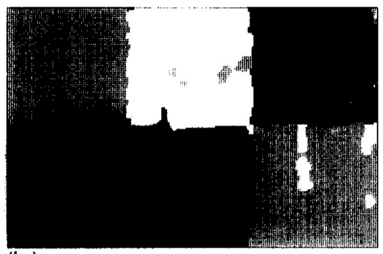


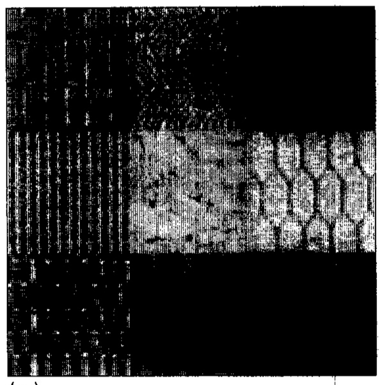


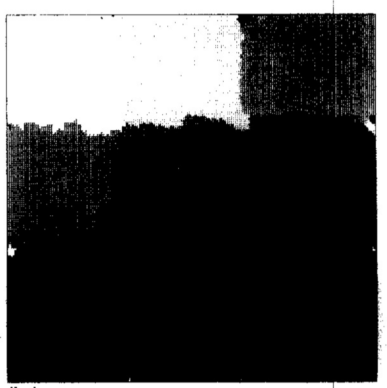


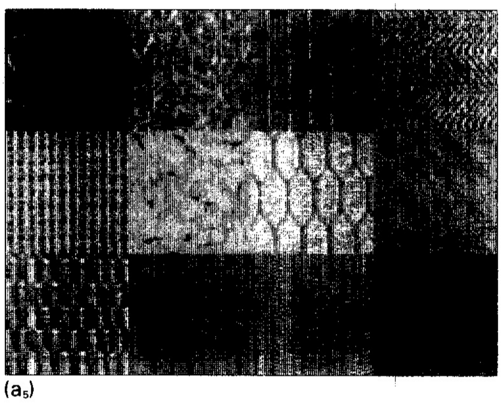












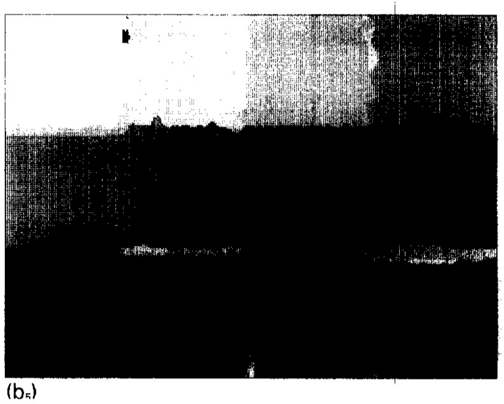


Fig. 6. (a_1) Mosaic of 4 textures; (a_2) mosaic of 5 textures; (a_3) mosaic of 6 textures; (a_4) mosaic of 9 textures; (a_5) mosaic of 12 textures; (b_1–b_5) the respective segmentation mappings of mosaics in (a_1–a_5).

the segmentation results using fractal geometry based features. Moreover, Hsiao and Sawchuk [24] reported an average classification accuracy of 88.1% for textures of size 128×128 and the results of the proposed segmentation technique, as discussed above, are slightly better.

In all of the simulation results described above we have used self-adaptive learning rates (with the initial values of η as 0.1) during training. That a fixed value of η (as in the original BP algorithm) fails to provide the desired result has been observed in a good number of segmentation tasks. During our extensive study of the present approach, it has been noted that if the value of the learning rate in the original BP algorithm is not set to a very small value (say, less than 0.1) the learning process often oscillates with respect to the system error. A smaller value of η (say, 0.01) may ensure convergence but becomes extremely

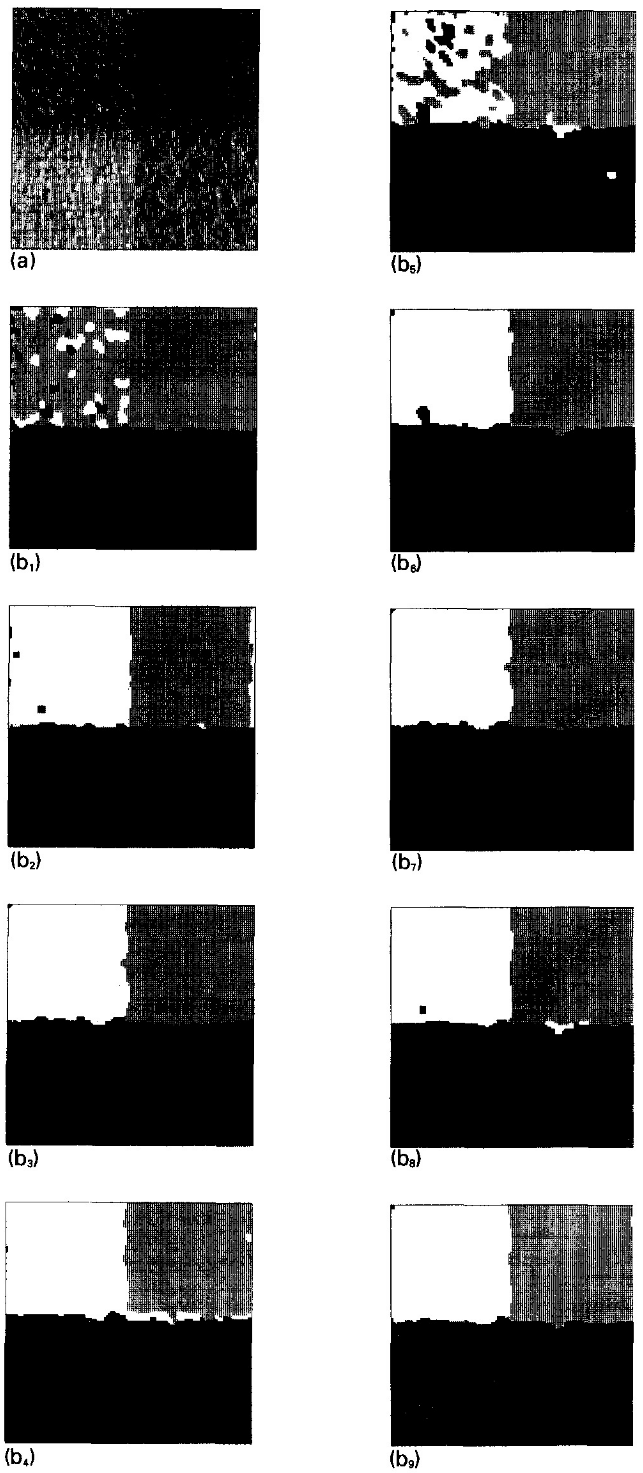


Fig. 7. (a) A mosaic of 4 textures; (b_1–b_4) the segmentation mappings corresponding to different window sizes as in Table 1; (b_5–b_9) the segmentation mappings corresponding to various hidden layer sizes as in Table 2.

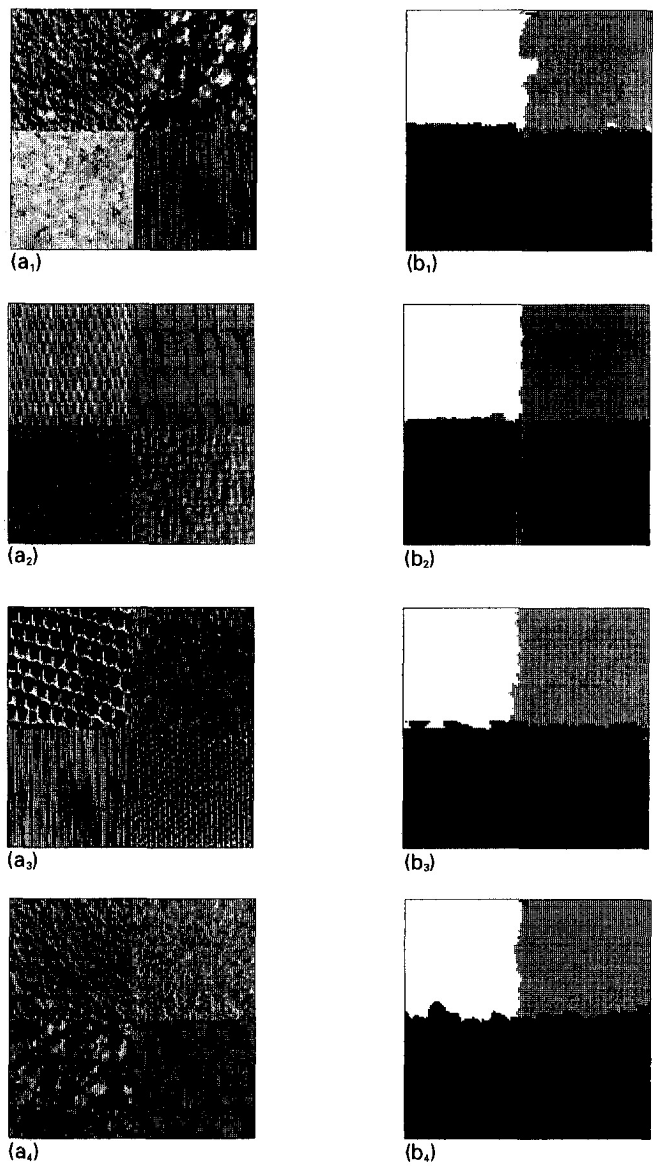


Fig. 8. (a_1–a_4) Four mosaics of different sets of four textures; (b_1–b_4) the corresponding segmentation mappings using the MLP-based method.

slow. This behavior of the BP algorithm has been reported earlier [25]. Moreover, we have compared the training performance of our BP algorithm using self-adaptive learning rates with that using a simple rule [25] for learning rate adaptation. We have seen that, in general, the BP algorithm with self-adaptive learning rates performs better in terms of convergence rate. The learning performance corresponding to these two different schemes of learning rate (L.R.) adaptation have been compared using texture classes in Fig. 7(a) and the results are plotted in Fig. 9.

5. Conclusions

The most interesting aspect of the proposed segmentation technique is that it does not require consideration of any feature set. The computational load of the training process in the proposed MLP-based segmentation method is significant. But once the network is trained, real-time segmentation is possible, whereas in the feature-based segmentation methods computation of features is time-consuming. Also, a modification of the BP algorithm has been proposed that can

Table 5
Learning and network statistics corresponding to segmentation tasks in Fig. 6

Texture mosaic	Input window size	No. of hidden nodes	No. of sweeps	Initial value of η	Final system error	% of correct classification
6(a_1)	11 × 11	12	215	0.1	0.075	95.56
6(a_2)	11 × 11	12	270	0.1	0.069	94.54
6(a_3)	11 × 11	20	320	0.1	0.077	94.45
6(a_4)	11 × 11	32	585	0.1	0.135	90.05
6(a_5)	15 × 15	44	1060	0.1	0.167	89.16

Table 6
Comparison of the proposed segmentation technique with an existing technique using fractal features

Texture mosaic	% of correct classification using fractal features					% of correct classification using proposed method
	1NN		5NN		Min-dist	
	A	B	A	B		
8(a_1)	87.00	87.00	85.81	86.90	85.28	95.72
8(a_2)	96.60	96.66	96.62	96.95	96.97	95.35
8(a_3)	94.68	94.68	94.65	94.72	94.68	93.82
8(a_4)	92.55	92.55	91.60	92.26	91.40	94.63
Average	92.71	92.72	92.17	92.71	92.08	94.88

efficiently avoid the convergence problem of the original algorithm in a real-life task such as texture segmentation. Finally, the robustness of the technique is established on the basis of the good number of examples considered in our simulation runs.

Universality is also an important aspect of the present technique. We have observed that minor pattern variations due to rotation, scaling or blurring do not have much effect on the segmentation results. For example, in Fig. 10 two textures have been rotated at different angles, blurred, stretched or scaled down using different factors, and the corresponding mosaics are shown in Fig. 10(a_1–a_4), respectively. The segmentation mappings are shown in Fig. 10(b_1–b_4). For these segmentation tasks, no separate training of the MLP network has been undertaken. The same network trained for the segmentation tasks of Fig. 5(a_1–a_3) has been used. Also, the segmentation results do not depend on the type of boundary between different textures. For example, for segmentation of Fig. 5(a_1–a_3), the network has been trained only once.

Acknowledgements

One of the authors (U. Bhattacharya) wishes to thank the University Grants Commission (UGC) of India for

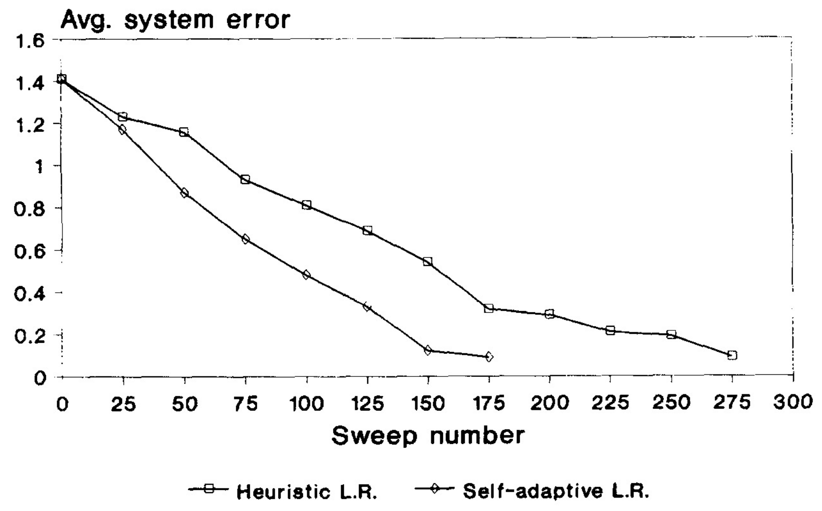


Fig. 9. The superior convergence performance of the training of the MLP-based texture classifier, when self-adaptive learning rates are used, compared to the use of learning rates adapted following certain rules based on heuristics (the segmentation task of Fig. 7(a) has been considered with the optimal choices of parameter values).

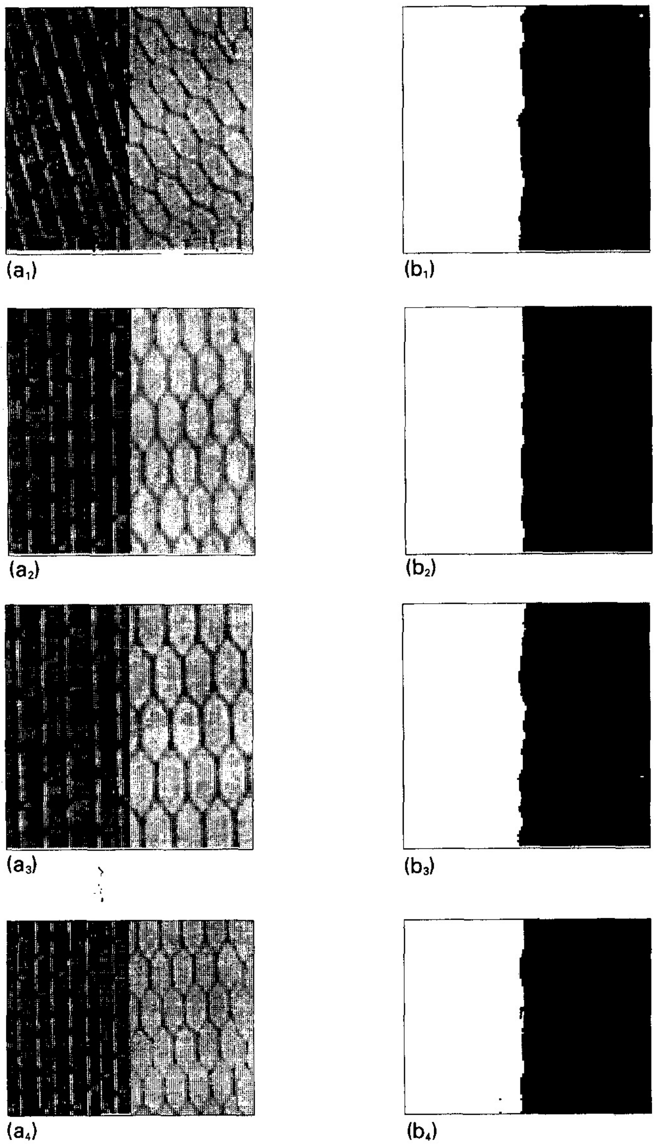


Fig. 10. (a_1) Four mosaics of two textures when the two textures have been rotated by 15° and 30°; (a_2) both textures have been blurred by mean filtering taking 3 × 3 windows; (a_3) the two textures have been stretched by 20% and 15%; (a_4) they have been scaled down by 20% and 15%; (b_1–b_4) the corresponding segmentation mappings using the MLP network trained for the segmentation tasks of Fig. 5(a_1–a_3).

providing him a research fellowship grant in pursuing his research work.

References

[1] R.M. Haralick, Statistical and structural approaches to texture, Proc. IEEE 67 (5) (1979) 786–804.
[2] L. v. Gool, P. Dewaele, A. Oosterlinck, Texture analysis anno 1983, Computer Vision, Graphics and Image Processing 29 (3) (1985) 336–357.
[3] M. Tuceryan, A.K. Jain, Texture analysis, in: C.H. Chen, L.F. Pau, P.S.P. Wang (Eds.), The Handbook of Pattern Recognition and Computer Vision, World Scientific Publishing, New York, 1992.
[4] T.R. Reed, J.M.H. du Buf, A review of recent texture segmentation and feature extraction techniques, Computer Vision, Graphics and Image Processing: Image Understanding 5 (3) (1993) 359–372.
[5] R.M. Haralick, K. Shanmugam, I. Dinstein, Textural features for image classification, IEEE Trans. Syst. Man Cybernetics SMC-3 (1) (1973) 610–621.
[6] P.C. Chen, T. Pavlidis, Segmentation by texture using co-occurrence matrix and split-and-merge algorithm, Computer Graphics and Image Processing 10 (2) (1979) 172–182.
[7] B.B. Chaudhuri, N. Sarkar, Texture segmentation using fractal dimension, IEEE Trans. Patt. Anal. Machine Intell. PAMI-17 (1) (1995) 72–77.
[8] J. Mao, A.K. Jain, Texture classification and segmentation using simultaneous autoregressive model, Patt. Recog. 25 (2) (1992) 173–188.
[9] B.S. Manjunath, R. Chellappa, Unsupervised texture segmentation using Markov random field models, IEEE Trans. Patt. Anal. Machine Intell. PAMI-13 (5) (1991) 478–482.
[10] C. Shang, K. Brown, Principal feature-based texture classification with neural networks, Patt. Recog. 27 (5) (1994) 675–687.
[11] F. Farrokhnia, Multi-channel filtering techniques for texture segmentation and surface quality inspection, Ph.D. Dissertation, Department of Electrical Engineering, Michigan State University, 1990.
[12] J.F. Haddon, J.F. Boyce, Neural networks for the texture classification of temporally consistent segmented regions of FLIR sequences, in: Proc. 12th IAPR Int. Conf. on Pattern Recognition, II, Jerusalem, Israel, 1994, pp. 107–111, IEEE Computer Society Press, 1994.
[13] S.R. Yhann, T.Y. Young, A multiresolution approach to texture segmentation using neural networks, 10th Int. Conf. on Pattern Recognition, 16-21 June 1990, Atlantic City, New Jersey, USA: Proceedings, I, IAPR, pp. 513–517.
[14] P. Brodatz, Textures: A Photographic Album for Artists and Designers, Dover, New York, 1966.
[15] D.E. Rumelhart, G.E. Hinton, R.J. Williams, Learning internal representations by error propagation, Institute for Cognitive Science Report 8506, University of California, San Diego, 1985.
[16] U. Bhattacharya, S.K. Parui, On the rate of convergence of perceptron learning, Patt. Recog. Lett. 16 (5) (1995) 491–497.
[17] R.A. Jacobs, Increased rates of convergence through learning rate adaptation, Neural Networks 1 (1988) 295–307.
[18] U. Bhattacharya, S.K. Parui, Self-adaptive learning rates in backpropagation algorithm improve its function approximation performance, 1995 IEEE Int. Conf. on Neural Networks: Proceedings, the University of Western Australia, Perth, Western Australia, 27 November-1 December 1995, Institute of Electrical and Electronic Engineers, New York, 1995, pp. 2784–2788.
[19] K. Hornik, M. Stinchcombe, H. White, Multilayer feedforward networks are universal approximators, Neural Networks 2 (1989) 359–366.
[20] J.K. Kruschke, Creating local and distributed bottlenecks in hidden layers of back-propagation networks, in: D. Touretzky, G. Hinton, T. Sejnowski (Eds.), Proc. 1988 Connectionist Models Summer School, San Mateo, Morgan Kaufmann, CA, 1988, pp. 120–126.
[21] M. Nagao, T. Matsuyama, Edge preserving smoothing, Computer Graphics and Image Processing 9 (1979) 394–407.
[22] J.J. Jiang, A.A. Sawchuk, Noise updating repeated weiner filter and other adaptive noise smoothing filters using local image statistics, Appl. Opt. 25 (1986) 2336–2337.
[23] B.B. Chaudhuri, N. Sarkar, P. Kundu, An improved fractal geometry based texture segmentation technique, Proc. IEE Part E 140 (1993) 233–241.
[24] J.Y. Hsiao, A.A. Sawchuk, Supervised textured image segmentation using feature smoothing and probabilistic relaxation techniques, IEEE Trans. Patt. Anal. Machine Intell. PAMI-11 (12) (1989) 1279–1292.
[25] T.P. Vogl, J.K. Mangis, A.K. Rigler, W.T. Zink, D.L. Alken, Accelerating convergence of the backpropagation method, Bio. Cybern. 59 (4/5) (1988) 257–263.