Texture Classification Using Logical Operators

Vidya Manian, Ramón Vásquez, *Senior Member, IEEE*, and Praveen Katiyar

***Abstract*—In this paper, a new algorithm for texture classification based on logical operators is presented. Operators constructed from logical building blocks are convolved with texture images. An optimal set of six operators are selected based on their texture discrimination ability. The responses are then converted to standard deviation matrices computed over a sliding window. Zonal sampling features are computed from these matrices. A feature selection process is applied and the new set of features are used for texture classification. Classification of several natural and synthetic texture images are presented demonstrating the excellent performance of the logical operator method. The computational superiority and classification accuracy of the algorithm is demonstrated by comparison with other popular methods. Experiments with different classifiers and feature normalization are also presented. The Euclidean distance classifier is found to perform best with this algorithm. The algorithm involves only convolutions and simple arithmetic in the various stages which allows faster implementations. The algorithm is applicable to different types of classification problems which is demonstrated by segmentation of remote sensing images, compressed and reconstructed images and industrial images.**

***Index Terms*—Image classification, logical operators, texture analysis, zonal filtering.**

I. Introduction

TEXTURE classification is an image processing technique by which different regions of an image are identified based on texture properties. This process plays an important role in many industrial, biomedical and remote sensing applications. Early work utilized statistical and structural methods for texture feature extraction [1]–[4]. Gaussian Markov random field (GMRF) and Gibbs distribution texture models were developed and used for texture recognition [5], [6]. Power spectral methods [1] using the Fourier spectrum have also been used. DCT, Walsh–Hadamard, and DHT have been used for recognition of two-dimensional binary patterns [7]. One of the major developments recently in texture segmentation has been the use of multiresolution and multichannel descriptions [8] of the texture images. This description provides information about the image contained in ever smaller regions of the frequency domain, and thus provides a powerful tool for the discrimination of similar textures. The use of scale-space-filtering is equivalent to a decomposition of the image in terms of wavelets. Several wavelet transform algorithms such as the pyramidal and tree structured wavelet transforms [9]–[12], Gabor filters [13], and the Haar [14] basis functions have been used for multiresolution and multichannel texture classification/segmentation. Laws [15] proposed a simple scheme which used local linear transformations and energy computation to extract texture features. This simple scheme often gives good results but is not consistent in performance. The statistical methods share one common weakness, of primarily focusing on the coupling between image pixels on a single scale and are also computationally intensive processes.

Logical operators have been used for Boolean analysis, minimization, spectral layered network decomposition, spectral translation synthesis, image coding, cryptography and communication. Logical systems considered in this work are logical Hadamard transform, adding and arithmetic transforms and logical operators such as equivalence, negation, and conjunction. A family of all essential RADIX-2 addition/subtraction transforms [16] has been developed. One of it is the well known Hadamard transform, the other is called arithmetic transform when applied to binary vectors. The third is the adding transform. The arithmetic and adding transforms are based on addition/subtraction of real numbers and are counterparts of generalized Reed Muller canonical expressions based on modulo-2 algebra. Fast computer implementation of these two transforms for logic design is presented in [17], and ways of generation of forward and inverse fast transform for orthogonal arithmetic and adding transform have been developed [18]. In [19], a new algorithm for computing Hadamard transform is presented. For simplicity, all the above logical systems are called operators in this work. Logical operators have been used recently for image compression [20]. As pointed out, fast algorithms [21] are already available for implementation of these schemes. But, surprisingly their usefulness in image classification has not been exploited. This is the first paper in open literature that applies the logical systems for applications other than in logical synthesis. This work is a unique attempt in the following respects:

1) construction of a texture feature space using logical operators;
2) the algorithm is computationally attractive with excellent performance over a wide variety of images.

This paper is organized as follows. Logical operators are described in Section II. In Section III, texture analysis using the operators is explained and the algorithm for texture classification is presented. In Section IV experimental results of classifying different types of images and comparison with other methods are presented. Section V presents application of the algorithm to segmentation problems. Finally, Section VI gives the conclusions and a few pointers on future directions.

Manuscript received February 4, 1999; revised May 10, 2000. This work was supported in part by the National Science Foundation under Grant CDA 9417659, NASA under Grants NCCW-0088 and NCC5340, and the Department of Electrical and Computer Engineering, University of Puerto Rico, Mayagüez. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Josiane B. Zerubia.

The authors are with the Department of Electrical and Computer Engineering, University of Puerto Rico, Mayagüez Campus, Puerto Rico (e-mail: reve@ece.uprm.edu).

Publisher Item Identifier S 1057-7149(00)07592-8.

$$\begin{bmatrix}1\\1\end{bmatrix}\begin{bmatrix}1\\-1\end{bmatrix}\begin{bmatrix}-1\\1\end{bmatrix}\begin{bmatrix}-1\\-1\end{bmatrix}\begin{bmatrix}1\\0\end{bmatrix}\begin{bmatrix}0\\1\end{bmatrix}\begin{bmatrix}0\\-1\end{bmatrix}\begin{bmatrix}-1\\0\end{bmatrix}\begin{bmatrix}0\\0\end{bmatrix}$$

Fig. 1. Basic operator generation matrices.

II. Logical Operators

The logical operators considered here are order-2 elementary matrices. The building blocks for defining these matrices are 0, 1, -1, matrices of order 1×1. The matrices can be formed by defining the following operations on these basic building blocks. This process of generation of order-2 matrices and operations have been described in [16].

A *row-wise join* (RWJ) or concatenation of a matrix A of order $n \times m$ and a matrix B of order $n \times m$ is the partitioned matrix of order $2\ n \times m$, such that its first n rows are exactly the same as the rows of matrix A and the rows from $n+1$ to $2n$ belong to matrix B. When this operation is applied to the above order 1×1 matrices for all possible concatenations, there are nine different matrices of order 2×1, which are shown in Fig. 1.

A *column-wise join* (CWJ) or concatenation of a matrix A of order $n \times m$ and a matrix B of order $n \times m$ is the partitioned matrix of order $n \times 2m$, such that its first n columns are exactly the same as the columns of matrix A and the columns from $n+1$ to $2n$ belong to matrix B. When this operation is applied to the order 2×1 matrices shown in Fig. 1, for all possible concatenations, there are 81 different matrices of order 2×2, some of which are shown in Fig. 2.

Generation of Higher Order Matrices: The Walsh functions in Hadamard order are generated when the standard Kronecker product of the elementary Hadamard matrix H_2 is performed with itself. Similarly, the arithmetic and adding operator matrices of higher orders are obtained by successive application of the Kronecker product to the core matrices shown in Fig. 2(b) and (c). When all these elementary matrices are denoted by the same symbol O_2, then higher order matrices are given by

$$O_N = (O_2)^{[n]} \tag{1}$$

where

- n exponent means the application of the Kronecker product n times;
- N order of the transform matrix;
- n $= \log_2 N$.

1) Logical Hadamard Operator: The logical Hadamard operator is nothing but the Walsh Hadamard transform (WHT) referred to in the literature [22]. Walsh functions and transforms [23] are important analytical tools for signal processing and have wide applications in digital communications, digital image processing, statistical analysis as well as in digital logic design. Because Walsh transforms are binary related, their generation and implementation is fairly simple. Global Walsh function generators have been built that produce three different ordered outputs, that is, natural (known as Hadamard), strict sequency (known as Walsh or Walsh–Karczmarz), and dyadic (known as Paley). These three Walsh transforms are symmetric, i.e., the inverse transform for each of these is the same as the forward transform with the accuracy to a constant coefficient. Besides these three symmetric Walsh transform there exists a nonsymmetric one, known as Radamacher–Walsh transform. In order to characterize Walsh functions, sequency is used which can be defined as sign changes per unit interval

$$seq = \begin{cases} \frac{1}{2}(nzc) & \text{for even } nzc \\ \frac{1}{2}(nzc+1) & \text{for odd } nzc \end{cases} \tag{2}$$

where nzc stands for the *number of zero crossings*. Sequency can be thought of as a generalization of frequency as it can be applied to functions whose zero-crossings may occur at irregular intervals and which may be aperiodic. The definition of sequency can be modified to include the discrete functions. Specifically, if the number of sign changes of a discrete function is defined as $k/2$ when k is even and $(k+1)/2$ when k is odd.

The choice of ordering depends on the particular application. However, only the Hadamard–Walsh matrix [24] has the recursive Kronecker product structure, and for this reason it is preferred over other possible variants of the Walsh transforms, like Walsh–Karczmarz, Radamacher–Walsh, and Walsh–Paley transforms. The WHT is based on a complete orthonormal set of rectangular functions known as Walsh functions. Its order-8 matrix is given by

Sequency

$$H_8 = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \end{bmatrix} \begin{bmatrix} 0 \\ 7 \\ 3 \\ 4 \\ 1 \\ 6 \\ 2 \\ 5 \end{bmatrix}. \tag{3}$$

2) Arithmetic Operator: In many of the applications of the arithmetic operator (AR), the values of only some spectral coefficients are needed. An efficient way [25] has been developed for calculating the transform, which has the ability to evaluate only some chosen spectral coefficients. Its order-8 matrix is given by

$$AR_8 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & -1 & -1 & 1 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & -1 & 0 & 0 & -1 & 1 & 0 & 0 \\ 1 & 0 & -1 & 0 & -1 & 0 & 1 & 0 \\ -1 & 1 & 1 & -1 & 1 & -1 & -1 & 1 \end{bmatrix}. \tag{4}$$

3) Adding Operator: Just like Arithmetic operator, Adding operator (AD) can also be expanded to higher order matrix using Kronecker expansion. Its order-8 matrix is given by

$$AD_8 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}. \tag{5}$$

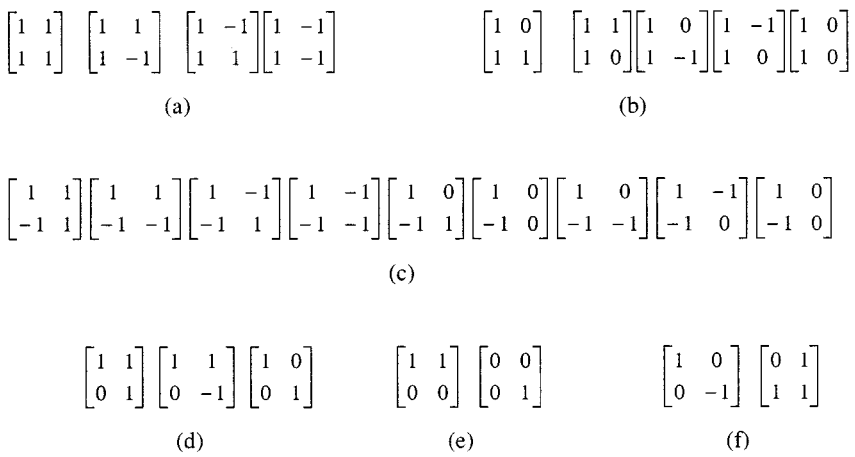


Fig. 2. Examples of generated operators (a) Hadamard, (b) adding, (c) arithmetic, (d) equivalence, (e) conjunction, and (f) disjunction.

4) Equivalence Operator "⇔": The equivalence operator [20] is true if x and y have identical values as shown in Fig. 3.

5) Conjunction Operator "∧": The conjunction of x and y, $x \wedge y$, is true if x and y are both true and is false otherwise. The conjunction operation is the same as the "AND" operation. It is a Boolean function of two arguments

$$C(x, y) = x \wedge y$$

$x\backslash y$	0	1
0	0	0
1	0	1

$x\backslash y$	0	1
0	CN	
1		

$$\text{where } CN = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}. \tag{6}$$

Conjunction operator [20], although not orthogonal, has been included because of its unique characteristic of retaining the original coefficients. If there is any information in the original image that can be used undistorted or before being transformed, then it can be exploited through this operator.

6) Disjunction Operator "∨": The disjunction of x and y, $x \vee y$, is false if x and y are both false and is true otherwise. The disjunction operation is the same as the "OR." Disjunction [20] is a Boolean function of two arguments:

$$D(x, y) = x \vee y$$

$x\backslash y$	0	1
0	0	1
1	1	1

$x\backslash y$	0	1
0	DN	
1		

$$\text{where } DN = \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}. \tag{7}$$

7) Negation Operator "¬": The negation [20] of x, $\neg x$, is false if x is true, and true if x is false. Negation operation performs the function of a "NOT." The equivalence, conjunction, disjunction and negation operators are summarized in Fig. 3.

x	Y	x⇔y	x ^ y	x v y	¬x
f	**f**	**t**	**f**	**f**	**t**
f	**t**	**f**	**f**	**t**	**t**
t	**f**	**f**	**f**	**t**	**f**
t	**t**	**t**	**t**	**t**	**f**

Fig. 3. Equivalence, conjunction, disjunction, and negation operations.

III. Texture Analysis and Classificaion with Logical Operators

The operators described in Section II can be exploited for their characteristic to relate texture elements or primitives in a logical context. Their ability to extract texture features and the algorithm for texture classification are presented below.

A. Texture Analysis

The operator masks are first convolved with texture regions

$$G(u, v) = F(u, v) * O(u, v) \tag{8}$$

where F is the image function and O is one of the set of logical operators. The response of the texture images to the six operators given in (8) is used to compute a standard deviation matrix using a sliding window

$$SD(u, v) = \frac{1}{W^2} \left\{ \sum_{m=-w}^{w} \sum_{n=-w}^{w} \times [G(u+m, v+n) - M(u+m, v+n)]^2 \right\}^{1/2} \tag{9}$$

where $W \times W$ is the size of the scanning window which is 5×5 and it slides pixel by pixel. M is the mean value of the window. The center pixel is assigned the standard deviation value. In order to avoid loosing boundary information the images are padded with zeros on all sides. Twelve images from the Brodatz album [26] with typical texture characteristics are

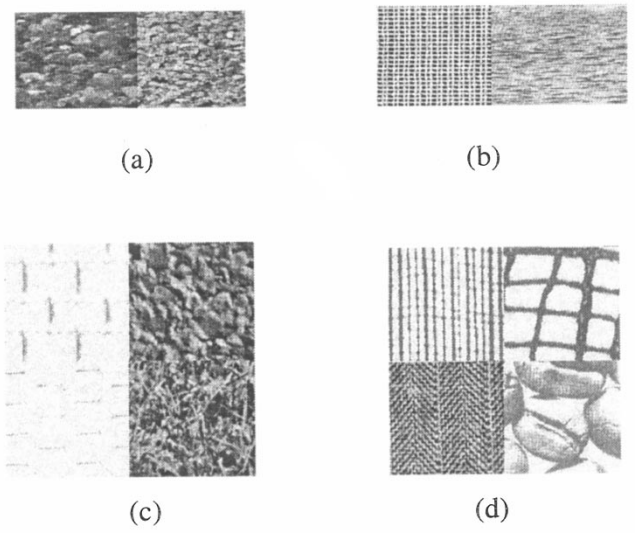


Fig. 4. Texture samples (a) coarseness, (b) contrast, (c) randomness, and (d) texture element size.

used (see Fig. 4). The operators in Fig. 2 are convolved with the texture samples of size 64×64, as per (8) which constitutes a filtering operation. The standard deviation matrix for each operator response is computed as in (9), which can be seen as a smoothing operation. Define ℓ to be:

$$\ell(A) = \frac{1}{N_1 N_2} \sum_{1}^{N_1} \sum_{1}^{N_2} |a_{ij}| \tag{10}$$

for any $N_1 \times N_2$ matrix A, (10) is a scaling of the vector ℓ_1 norm

$$e = \ell(SD(u, v)). \tag{11}$$

From the scaled ℓ_1 norm values the properties of the operators can be examined. It is known that the Hadamard operators have good energy compaction properties. It can also be verified that they yield maximum discrimination for coarse and fine textures as can be seen from the values of (11) [see Fig. 5(a)] for the four Hadamard operators (H1–H4) for textures d28 and d29 [shown on left and right of Fig. 4(a)]. The values have been normalized, 1 represents maximum coarseness. The adding and arithmetic operators (AD1–AD5 and AR1–AR9) extract contrast properties from the textures with a value close to 1 for high contrast textures. High and low contrast textures d21 and d38 are shown in the left and right of Fig. 4(b), respectively. The corresponding values are plotted in Fig. 5(b) which shows that highest separation is obtained with operators AD1 and AR5. The conjunction and disjunction operators (CN1–CN2 and DN1–DN2) extract randomness information from the texture and have higher values close to 1 for irregular or random textures [d4 and d9 shown in right top and bottom of Fig. 4(c)], and lesser values for regular and periodic textures [d6 and d34 shown in left top and bottom of Fig. 4(c)]. As seen from the plot in Fig. 5(c), CN1 and DN1 operators yield maximum separability. The equivalence operators (EQ1–EQ3) perform an averaging operation over the textured image, and give a cue of the size of the texture element yielding higher values for smaller elements [textures d105 and

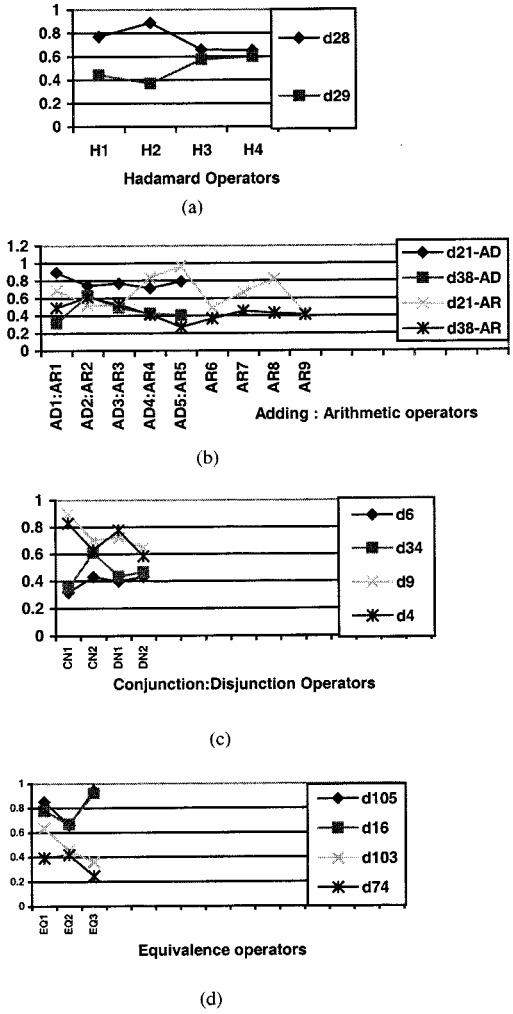


Fig. 5. Operator response plots.

$$\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \quad \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} \quad \begin{bmatrix} 1 & 0 \\ -1 & 1 \end{bmatrix}$$

O_1 : Hadamard O_2 : Adding O_3 : Arithmetic

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \quad \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} \quad \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}$$

O_4: Equivalence O_5: Conjunction O_6: Disjunction

Fig. 6. Selected operators for texture classification.

d16 shown in left top and bottom of Fig. 4(d)] and lesser values for larger texture structures [textures d103 and d74 shown in right top and bottom of Fig. 4(d)]. The graph in Fig. 5(d) shows maximum discrimination with the equivalence operator EQ3. None of the operators considered here capture directionality. From this analysis, one operator from each class of logical operator that gives the maximum separability among textures is chosen as the most powerful among the rest. The final set of six operators (H2, AD1, AR5, EQ3, CN1, and DN1) is shown in Fig. 6.

(a) (b)

Fig. 7. (a) French canvas texture sample and (b) SD matrix.

B. Algorithm for Texture Classification

1) Feature Extraction Process: The texture samples are convolved with the operators as in (8). The standard deviation matrix for each response is computed as in (9). All the operators are of size 2×2, which is adequate in generating an efficient feature space. Features are extracted by zonal-filtering using zonal masks [15], [27] which are applied to the standard deviation matrix, $SD(u, v)$, where $1 \leq u \leq N_1$ and $1 \leq v \leq N_2$, N_1 and N_2 are the number of rows and columns in the matrix [Note: $SD(u, v)$ represents the transformed domain and not the frequency domain]. A sample texture and its SD matrix for the adding operator response are shown in Fig. 7(a) and (b), respectively. The zonal mask, also called zonal filter, is a simple slit/mask or an aperture (see Fig. 8). A combination of an angular slit with a bandlimited low-pass, band-pass or high-pass filter can be used for yielding good discriminating features for periodic or quasiperiodic textures. Masks are sets of integers that are used to extract features from the standard deviation matrix.

Horizontal slit feature

$$Y_1 = \sum_{(u, v) \in H_m} SD(u, v) \tag{12}$$

where the horizontal slit mask $H_m = \{(u, v)\colon u, v \text{ integer}, U_1 \leq u \leq U_2;\ \leq 1 \leq v \leq N_2\}$

Vertical slit feature:

$$Y_2 = \sum_{(u, v) \in V_m} SD(u, v) \tag{13}$$

where the vertical slit mask $V_m = \{(u, v)\colon u, v \text{ integer}, 1 \leq u \leq N_1;\ V_1 \leq v \leq V_2\}$ If x and y are defined as $x = c + dv;\ y = a + bu$, where

$$c = -\frac{N_2 + 1}{N_2 - 1}, \quad d = \frac{2}{N_2 - 1}, \quad a = \frac{N_1 + 1}{N_1 - 1}$$
$$\text{and} \quad b = -\frac{2}{N_1 - 1}.$$

Then, ρ and θ are in polar coordinates and are defined as $\rho(u, v) = \sqrt{(a + bv)^2 + (c + du)^2}$ and $\tan \theta(u, v) = ((a + bu)/(c + dv))$

Ring feature:

$$Y_3 = \sum_{(u, v) \in R_m} SD(u, v) \tag{14}$$

where the ring mask $R_m = \{(u, v)\colon u, v \text{ integer}, \rho_1 \leq \rho(u, v) \leq \rho_2\}$

Circular feature:

$$Y_4 = \sum_{(u, v) \in C_m} SD(u, v), \tag{15}$$

where circular mask $R_m = \{(u, v)\colon u, v \text{ integer}, \rho_1 \leq \rho(u, v) \leq \rho_2\}$

Sector feature:

$$Y_5 = \sum_{(u, v) \in C_m} SD(u, v), \tag{16}$$

where $S_m = \{(u, v)\colon u, v \text{ integer}, \theta_1 \leq \theta \leq \theta_2\}$ These features can be computed with different sizes of masks and they form a feature vector Y for samples from each texture class.

2) Normalization of Features: Normalization of feature values is necessary to give equal weight to different features, especially when distance classifiers are used. Normalization also reduces the number of computations at the classification stage. Let the length of the feature vector Y be L, $\hat{Y}$ is the normalized resulting vector, i is the index for Y, such that $i \leq L$ then different types of normalization strategies are as summarized in Table I.

3) Feature Selection: A combination of two criteria, the distance between the means of each feature and the measure of standard deviation are used to quantify the separation between classes. If M is the mean and σ is the standard deviation of the features in the training matrix, the sum of the distances of each feature from M and σ is computed as

$$D_L = \sum_j \left| \hat{Y}_{i, j} - M_i \right| \tag{17}$$

where

i	feature index;
L	number of features;
j	class index;
J	total number of texture classes.

The standard deviation value for each feature is computed as

$$\sigma_L = \sqrt{\frac{1}{J} \sum_{j=1}^{J} (Y_{i, j} - M_i)^2}. \tag{18}$$

Values computed from (17)–(18) are sorted in ascending order and the features corresponding to the first half of the sequence are selected as the best overall set of features as shown in Fig. 9. These measures are referred to as measures of separability which implies the ease with which patterns can be correctly associated with their classes by means of statistical pattern classification.

4) Classification: Both the parametric and nonparametric classifiers are used in the experiments. The Bayes classifier, Euclidean, and Mahalanobis distance measures are used. K-Nearest Neighbor (k-NN), and "leave-one-out" classification methods have been used [28]. The different classifier measures

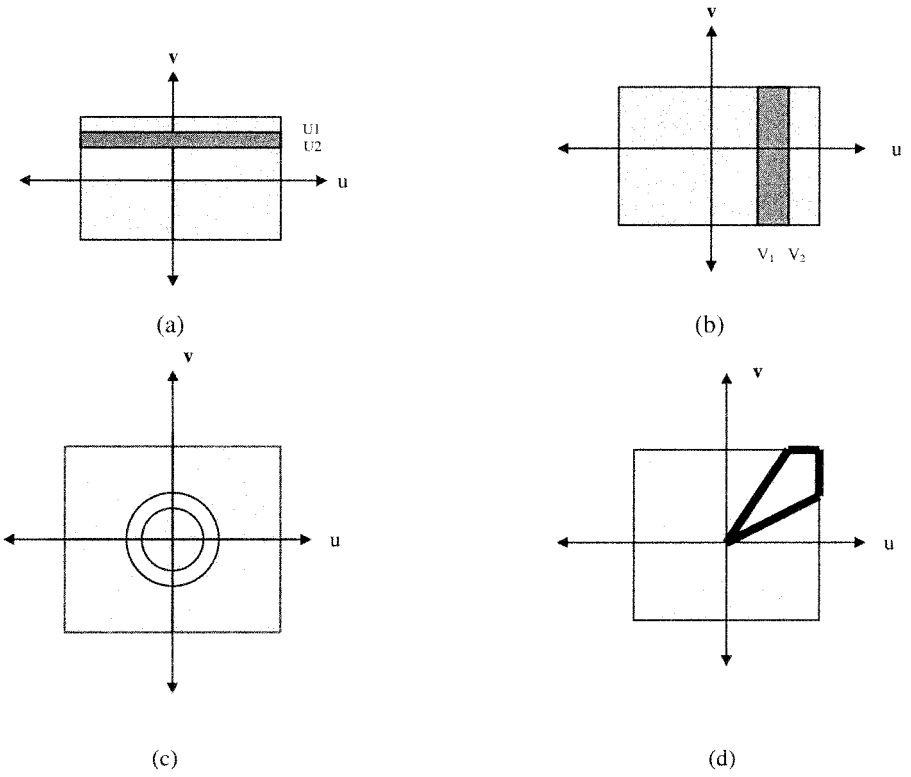


Fig. 8. Zonal masks (a) horizontal mask, (b) vertical mask, (c) ring mask, and (d) sector mask.

TABLE I
NORMALIZATION

Type 1	$\hat{Y} = \frac{Y_i}{Y_{\max}} \quad \forall i$	
Type 2	$\hat{Y} = \frac{Y_i - E\{Y\}}{\sigma_Y}$	
Type 3	$\hat{Y} = \frac{Y_i}{\sum_{i=1}^{L} \lvert Y \rvert} \quad \forall i$	
Type 4	$\hat{Y} = \frac{Y_i}{\lvert Y_i \rvert_{\max}} \quad \forall i$	*(same as Type 1 in this context)*

are summarized in Table II. C_j is the covariance matrix for the feature set of texture j. The Bayes classifier is the optimal case, where the features are assumed to have a Gaussian density function. The maximum likelihood rule is obtained from the Bayes classifier [29] when no useful information is provided for $P(w_j)$ which is the probability of occurrence of class w_j. When all class covariances are equal, the term ln $|C_i|$ in the maximum likelihood estimate is not discriminating and hence ignored. This results in the Mahalanobis distance classifier [29]. As this is advantageous and faster over the maximum likelihood classifier, it is used here. In cases where the covariance estimate is not accurate, a classifier that depends only on the mean positions of the texture classes can be used. This is the case of the minimum Euclidean distance classifier where the training data is used only to determine the class means; classification is performed by placing the unknown feature vector in the class of the nearest mean. In the k-NN case the unknown sample is classified by assigning it the label most frequently represented among the k nearest samples. Generally, a training data set is used to train or design a classifier and another testing data set is used to test the classifier. There are more options if this process can be repeated several times. In leave-one-out classification, one sample is excluded and the classifier is designed on the remaining $N - 1$ samples each time, and the excluded sample is tested by the classifier. This classifier involves heavy computational costs.

IV. Experimental Results

This section presents experiments that test the recognition capability of the algorithm and compares its performance with other methods. The effect of important parameters such as type of normalization of features and type of classifier are also discussed.

A. Classification Using Brodatz Textures

The complete algorithm for texture discrimination is shown in Fig. 9. It is used to classify textures from the Brodatz album. Thirty–three different textures are used in this experiment. Each image is of size 256×256 pixels scanned at 300 dpi resolution with 256 gray levels. The experimental setup is described below.

- *Training Phase*
  1) One–hundred twenty-eight training samples each of size 64×64 are extracted using overlapping blocks from each texture image. This size is chosen based on texture structure.
  2) One sample is convolved with the logical Hadamard operator.
  3) The standard deviation matrix is computed on the convolution response.

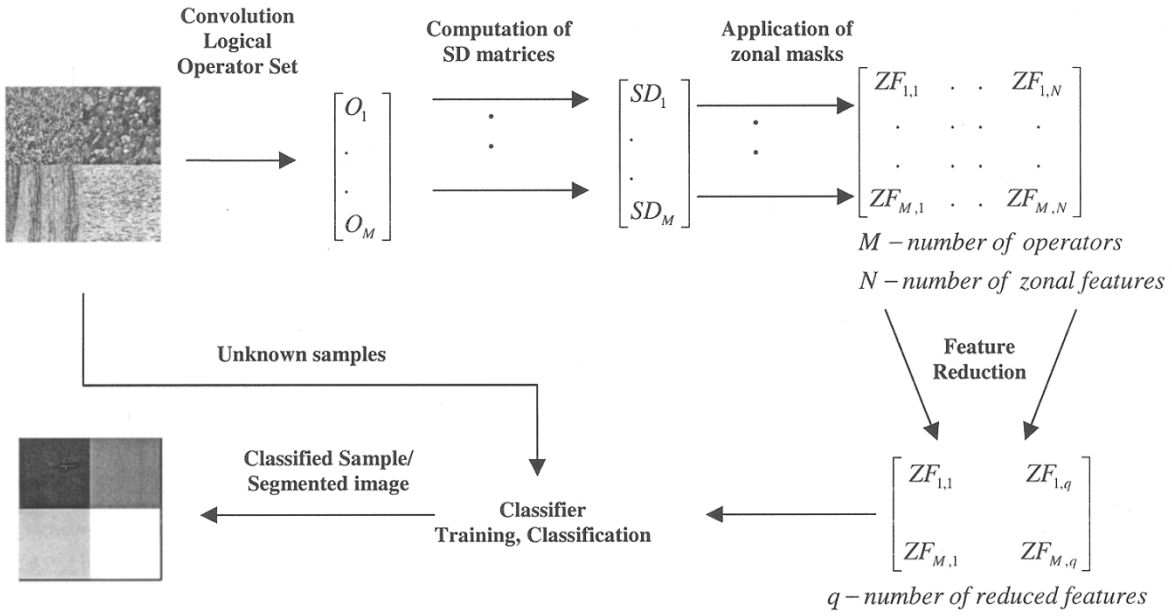


Fig. 9. Algorithm for texture classification.

TABLE II
CLASSIFIER MEASURES

1. Bayes : $d_j(x) = \ln P(w_j) - \frac{1}{2}\ln|C_j| - \frac{1}{2}\left[(x-m_j)^T C_j^{-1}(x-m_j)\right]$

2. Mahalanobis distance: $d_j(x) = \left\{(x-m_j)^T C_j^{-1}(x-m_j)\right\}$

3. Minimum Euclidean distance: $d_j(x) = \sum_{q=1}^{Q}\left(x_q - m_{i,q}\right)^2$

4. k-Nearest Neighbor: $P(w_j \mid x)$ if $k-nearest\ neighbors$ of x are labeled w_j

5. Leave one out:

$$d_j(x) = \sum_{q=1}^{Q}|x - x_{N-1}| \text{ where N is the total number of test samples}$$

4) Zonal filtering masks are applied to the standard deviation matrices and features are computed.
5) Steps 2–4 are repeated for the adding, arithmetic, equivalence, conjunction and disjunction operators.
6) Steps 2–5 are repeated for the fixed number of training samples (128 in this case).
7) Steps 1–6 are repeated for the 33 classes of textures.

- *Classification Phase*

8) A texture sample from the first textured image which is to be classified is selected.
9) Test feature vector is computed as in steps 2–5.
10) The feature vector is normalized.
11) Feature space reduction algorithm is applied.
12) A classifier is used to identify the unknown sample.
13) Steps 8–12 are repeated for 128 distinct testing samples taken from a set of 33 testing images different from the training images.

The above steps of the experiment has been repeated by applying each of the four types of normalizations, and classification has been done with five different classifiers. The number of features selected on an average is nine. The best normalization type and the best classifier along with the average percentage of correct classification (PCC) for each texture are tabulated in Table III. One–hundred twenty-eight training samples and 128 testing samples are used; 100% PCC is obtained with all training samples. The results in Table III show PCC for the testing samples.

TABLE III
BEST CLASSIFICATION RESULTS FOR INDIVIDUAL TEXTURES

Texture ID	Texture Name	PCC	Norm-alization Type	Classifier Used
D-94	Brick	100	4	k-NN
D-101	Cane	97	4	k-NN
D-36	Skin	100	4	k-NN
D-84	Raffia	72	4	k-NN
D-56	Strawmat	98	4	Euclidean
D-22	Skin	100	4	Euclidean
D-28	Sand	100	4	Euclidean
D-9	Grass	100	4	Euclidean
D-37	Water	100	4	Euclidean
D-4	Cork	100	4	Euclidean
D-57	Paper	100	4	Euclidean
D-50	Raffia-2	100	4	Euclidean
D-20	Canvas	100	4	Euclidean
D-6	Wire	100	4	Euclidean
D-105	Cloth-1	100	4	Euclidean
D-24	Leather	100	4	Euclidean
D-68	Wood	100	4	Euclidean
D-77	Canvas	100	2	Euclidean
D-16	Weave	100	2	Euclidean
D-15	Straw	89	2	Euclidean
D-90	Clouds	100	4	Euclidean
D-74	Coffee	73	4	Euclidean
D-93	Fur	100	4	Euclidean
D-34	Netting	100	4	Euclidean
D-65	Rattan	84	4	Euclidean
D-53	Cloth-2	100	4	Euclidean
D-82	Cloth-3	86	0	Euclidean
D-52	Cloth-4	81	0	Euclidean
D-19	Wool	97	0	Euclidean
D-78	Cloth-5	95	4	Euclidean
D-103	Burlap	100	4	Euclidean
D-12	Bark	100	4	Euclidean
D-79	Cloth-6	92	4	Euclidean

B. Comparison with Other Methods

In order to significantly establish the superiority of the logical operator method, its performance is compared with the most popular techniques in texture classification from the wide spectrum of methods available. The spatial gray level dependence method (SGLDM) or co-occurrence matrix method, Fourier power spectrum method, tree-structured wavelet transform method, Laws texture features and Gabor method are used. The SGLDM estimates the second-order joint conditional probability density functions, $f(i,j|d,\theta)$, $\theta = 0°$, $45°$, $90°$, $135°$, written in matrix form and are called co-occurrence matrices. Haralick proposed [3] 14 statistical features that can be computed from these matrices. Although SGLDM has been proven to perform well for texture classification, the selection of the appropriate distance between pixels and angle for the co-occurrence matrix computation poses a problem and it is also computationally intensive. The Fourier spectrum method has not performed well even in earlier comparisons [1], [2]. The statistical Fourier features of average magnitude, maximum magnitude, energy and features using the zonal masks used in [1], [2] are computed. In the tree-structured wavelet transform (TWT) method, the texture samples are decomposed into a multiresolution hierarchy only at nodes where the energy of the decomposed subimage is not significantly smaller than the other subimages at that level. The energy map of the channels is used as a feature vector for classification [9]. Energy features up to four levels of decompositions are considered. This method has the drawback of becoming noisy at higher levels of decompositions. Four of the most powerful Laws masks are used to compute the texture energy measure [30]. As the masks are ideal, when the textures are complex they perform poorly. The Gaussian window function is used to compute the Gabor transform and the Gabor coefficients are approximated using an optimization criteria [31]. Average energy and residual features are computed with this method. As the Gabor coefficients are not accurate, they do not sufficiently discriminate the textures. The feature selection process is applied with all methods in order to obtain the best feature set for each algorithm. The Euclidean distance classifier is used in all the experiments for the comparisons. Six different experiments with six texture classes in each case are conducted. The resulting PCCs are given in Table IV. As can be seen, the logical operator method outperforms the others with an average PCC of 93%.

C. Effect of Different Classifiers and Normalization Techniques

Different classifiers are used with the logical operator method, to choose the best classification technique with this algorithm. Nine experiments with six textures each were conducted with each type of classifier mentioned in Table II. The PCCs for this experiment are shown in Table V. All the classifiers perform close to each other. An original set of 24 features using the masks in Fig. 8 is computed. After feature reduction, the average number of features selected is seven. The best features are found to be those obtained from the slit masks in Fig. 8(a) and (b) and sector mask in Fig. 8(d). The Euclidean distance measure is found to be the most suitable one for this algorithm due to the nature of features used, the classifier simplicity and speed. This classifier gives the best performance which is also evident from the results in Table I. Also, as seen from this table, the best type of feature normalization is found to be Type 4. Type 0 refers to no normalization.

TABLE IV
COMPARISON WITH SGLDM, FPS,TWT, LAWS, AND GABOR METHODS

Textured Image	% Correct Classification (PCC) results with different classifiers					
Mosaic of six Brodatz Textures with Texture ID's	Logical Transform	SGLDM	FPS	Wavelet algorithm	LAWS	Gabor algorithm
D-94/101/36/84/103/56	93	67	54	63	52	62
D-28/20/9/38/50/57	96	84	75	62	84	70
D-90/74/93/34/65/53	88	64	56	62	47	67
D-105/79/82/52/19/78	89	59	54	53	46	55
D-28/9/57/24/4/38	99	67	56	65	77	63
D-103/105/12/78/79/82	90	77	56	58	58	59
Overall PCC	93	70	59	61	61	63

TABLE V
COMPARISON OF DIFFERENT CLASSIFIERS

Textured Image	% Correct Classification (PCC) results with different classifiers				
Mosaic of six Brodatz Textures with Texture ID's	k-nearest neighbor classifier	Euclidean Distance classifier	Leave-one-out classifier	Bayes Classifier	Mahalanobis Classifier
D-94/101/36/84/103/56	93	93	92	71	82
D-22/28/9/38/4/57	99	100	100	79	88
D-28/20/9/38/50/57	96	99	96	84	96
D-6/105/24/19/68/16	97	98	98	71	72
D-77/4/16/15/24/9	94	97	96	75	82
D-90/74/93/34/65/53	88	93	87	74	79
D-105/79/82/52/19/78	89	91	83	75	83
D-28/9/57/24/4/38	99	99	97	80	86
D-103/105/12/78/79/82	90	95	93	92	93
Average PCC	94	96	94	78	85

V. APPLICATIONS

Different segmentation problems are presented in this Section, to emulate real world situations and evaluate the performance of the algorithm in these applications. The segmenting of remote sensing images, images subjected to compression techniques and industrial images such as textiles are considered.

A. Segmentation of Remote Sensing Images

The first image is an original SIR-C/X-SAR image of Manaus, Brazil, shown in Fig. 10(a). The segmentation using the logical operator algorithm into three distinct regions of flooded forests/shrubs, unflooded fields/forests and open water is shown in Fig. 10(b). The second image is an aerial photograph of a coastal region in Puerto Rico shown in Fig. 11(a), which has three distinct regions of sea, land and highway. Both images are of size 256×256. The algorithm procedure is the same described in Section III-B. The number of features selected for both images are 4. The window size used in this application is 8×8 in order to capture the texture characteristics of small regions and obtain an exact segmentation. The classification is done using this moving window with an overlap of seven pixels in order to avoid blocking effect. The segmentation has been compared with that of the co-occurrence method to estimate the performance and computational speed of the algorithm. Fig. 11(b) and (c) shows the segmentation results using the logical operator and the co-occurrence matrix

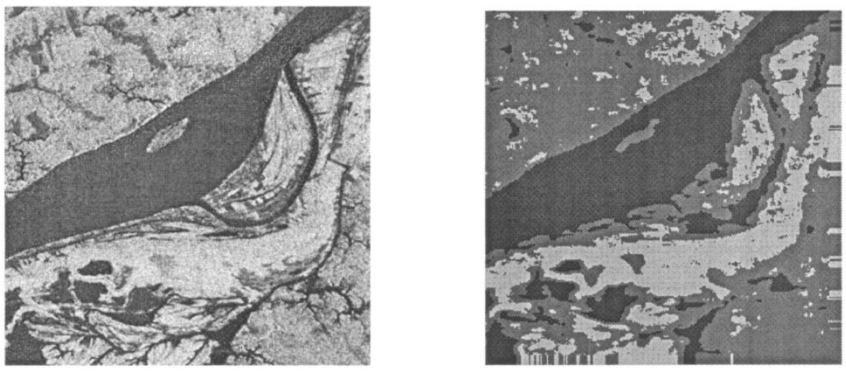


Fig. 10. (a) SIR-C/X-SAR image of Manaus, Brazil, and (b) segmented image using logical operator algorithm.

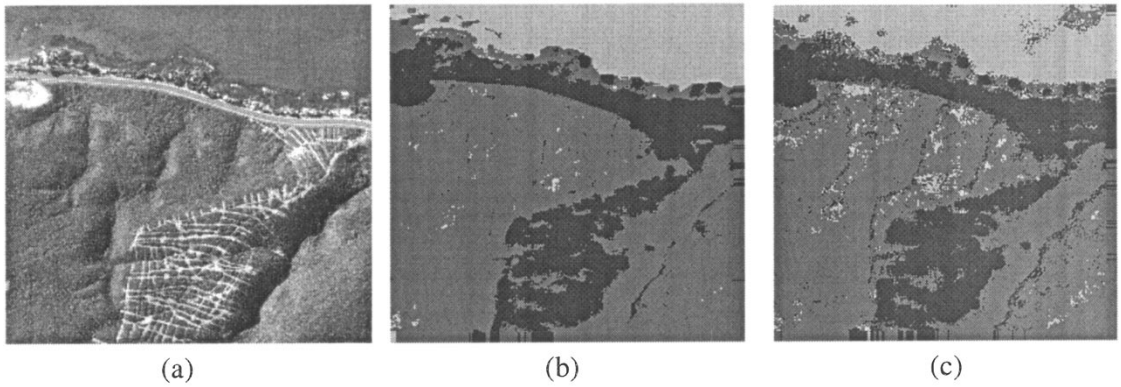


Fig. 11. (a) Original coastline image, (b) segmented image using logical operator algorithm, and (c) segmented image using co-occurrence method.

TABLE VI
TIMING RESULTS

	Time (x 10^3 sec.)	
	SGLDM method	Logical operator method
Segmenting image in Fig. 10(a)	16.59	1.2764
Segmenting image in Fig. 11(a)	24.8042	1.809

methods, respectively. Out of the total 65 536 pixels, 5898 pixels are mismatched using the logical operator algorithm giving a 9% error rate and 11 201 pixels are mismatched using the co-occurrence method, which gives an error rate of 17%. This result and a careful visual examination shows the superiority of the logical operator algorithm. Also, the logical operator method is much faster compared to the co-occurrence method. Table VI gives the timing results for performing the segmentation of the images in Figs. 10(a) and 11(a) on a Sun Ultra 2 workstation using the SGLDM and logical operator methods. The logical operator is on an average 13 times faster than the SGLDM algorithm in these experiments.

B. Segmentation of Compressed Images

The image shown in Fig. 11(a) has been compressed using the Hadamard transform, the JPEG quantization and coding schemes. The compression ratio (CR) obtained is 21 : 1 and the peak signal-to-noise ratio (PSNR) is 23. The reconstructed image is shown in Fig. 12(a). The segmented image using the logical operator algorithm is shown in Fig. 12(b). The number of features selected are 5. Because of the excessive blocking effect, the texture regions in Fig. 12(a) are corrupted. However, on comparing with the segmentation in Fig. 11(b), the logical operator algorithm has correctly assigned most of the texture regions of the reconstructed image to the correct class.

C. Segmentation of Textile Images

The original mosaic of six textile textures (basket, naugahyde leather, a fabric texture, corduroy, cotton, and tanned leather) is shown in Fig. 13(a). Each texture is of size 256×256 and the mosaic is of size 512×768. The algorithm is applied to segment this image using a moving window of size 8×8 with an overlap of seven pixels. The segmentation result is shown in Fig. 13(b) which shows excellent classification with minimal errors in the boundary proving the suitability of the algorithm for industrial applications.

D. Discussions

The individual results for the classification of 33 Brodatz textures are summarized in Table VII; 100% correct classification has been obtained for 22 textures and the PCC for the remaining textures are above 84. In general, the PCC increases while increasing the number of training samples up to 96 and remains constant for further increase [32]. The sample window size depends on the texture structure. In the experiments with Brodatz textures (Section III-A and IV), texture samples of 64×64 size

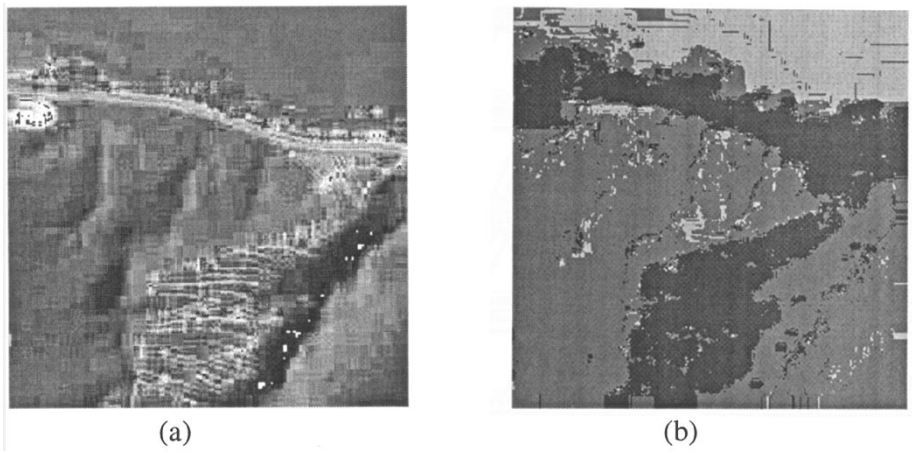


Fig. 12. (a) Reconstructed image after compression and (b) segmented image

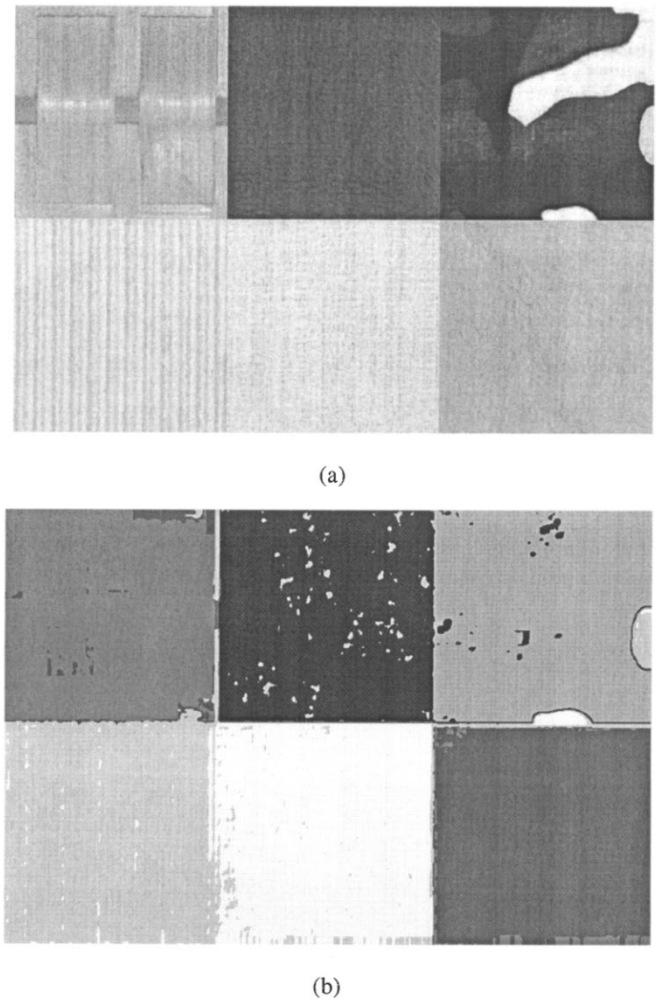


Fig. 13. (a) Original textile composite image and (b) segmented image

is optimal due to the wide range of textures involved with small to large structures. In general, the sample size should be large enough to yield reliable features and small enough to produce accurate boundaries in segmentation problems. Hence, 8×8 window size has been used in segmentation of remote sensing images shown in Figs. 9(a) and 10(a), where texture structure to be characterized are small. To avoid the curse of dimensionality, a feature selection process has been applied to select the optimal set of features.

TABLE VII
OVERALL PCC RANGE FOR ALL TEXTURES

% correct classification range	Number of Textures Correctly classified
100	22
95-99	4
90-94	1
85-89	2
< 84	4

VI. CONCLUSIONS

An efficient algorithm based on logical operators has been developed and proved to give excellent results with different types of texture images. Based on the ability to differentiate texture characteristics, six operators are chosen as the best in yielding discriminating features. Operator size of 2×2 is sufficient in transforming the textures in to an effective feature space. Higher order operators do not improve classifications and are also computationally more expensive. The features are computed from simple zonal filtering. The experiments and comparisons presented prove the robustness and versatility of the algorithm.

This method has an edge over other methods in that it can be implemented using fast algorithms that have already been developed for logical operators. The algorithm involves convolutions with 1, -1, and 0, which can be implemented efficiently and the feature extraction and classification stages involve only additions and comparisons. Any suitable hardware or logical structure such as DSP processors or FPGA can be utilized. The algorithm permits development of many ways of parallelization and hence it can be implemented in parallel computers or in distributed systems.

Different applications were presented, one of which is the segmentation of a compressed image. This is a particular attempt in expanding the scope of any texture classification technique. The logical operator algorithm clearly has greater potential than any other method in the aspect of efficiency, implementation and application areas. As a future direction, the capability of the algorithm for classifying binary images can be investigated, in line with the binary field transform (BFT), which has

been used for image compression [33]. The algorithm should be employed for classifying other types of images such as medical images and for object recognition.

REFERENCES

[1] R. W. Conners, "A theoretical comparison of texture algorithms," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-2, pp. 204–222, May 1980.

[2] J. S. Weszka, C. R. Dyer, and A. Rosenfeld, "A comparative study of texture measures for terrain classification," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-6, pp. 269–285, Apr. 1976.

[3] R. M. Haralick, K. Shunmugam, and I. Dinstein, "Textural features for image classification," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-3, pp. 610–621, Nov. 1973.

[4] T. M. Caelli, "An adaptive computational model for texture segmentation," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-18, pp. 9–17, 1988.

[5] A. Speis and G. Healey, "An analytical and experimental study of the performance of Markov random fields applied to textured images using small samples," *IEEE. Trans. Image Processing*, vol. 5, pp. 447–458, Mar. 1996.

[6] S. Krishnamachari and R. Chellappa, "Multiresolution Gauss–Markov random field models for texture segmentation," *IEEE. Trans. Image Processing* , vol. 6, pp. 251–267, Feb. 1997.

[7] J. Wu and W. Duh, "Feature extraction capability of some discrete transforms," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-36, p. 1687, Oct. 1991.

[8] A. C. Bovik, M. Clark, and W. S. Geisler, "Multichannel texture analysis using localized spatial filters," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 12, pp. 55–73, Jan. 1990.

[9] T. Chang and C.-C. J. Kuo, "Texture analysis and classification with tree-structured wavelet transform," *IEEE Trans. Image Processing*, vol. 2, pp. 429–441, 1993.

[10] C. S. Lu, P. C. Chung, and C. F. Chen, "Unsupervised texture segmentation via wavelet transform," *Pattern Recognit.*, vol. 30, pp. 729–742, 1997.

[11] E. Salari and Z. Ling, "Texture segmentation using hierarchical wavelet decomposition," *Pattern Recognit.*, vol. 28, pp. 1819–1824, 1995.

[12] V. Manian and R. Vásquez, "Scaled and rotated texture classification using a class of basis functions," *Pattern Recognit.*, vol. 31, pp. 1937–1948, Dec. 1998.

[13] O. Pichler, A. Teuner, and B. J. Hosticka, "A comparison of texture feature extraction using adaptive Gabor filtering, pyramidal and tree structured wavelet transforms," *Pattern Recognit.*, vol. 29, pp. 733–742, 1996.

[14] T. Lonnestad, "A new set of texture features based on the Haar transform," *11th Int. Conf. Acoustics, Speech, Signal Processing*, vol. 4, pp. 661–664, 1992.

[15] W. K. Pratt, *Digital Image Processing*. New York: Wiley, 1991.

[16] B. J. Falkowski and M. A. Perkowski, "A family of all essential Radix addition/subtraction multipolarity transforms: Algorithms and interpretations in Boolean domain," in *Proc. 23rd IEEE Int. Symp. Circuits Systems*, New Orleans, LA, May 1990, pp. 1596–1599.

[17] I. Schaefer, B. J. Falkowski, and M. A. Perkowski, "A fast computer implementation of adding and arithmetic multipolarity transforms for logic design," in *Proc. 34th IEEE Midwest Symp. Circuits Systems*, Monterey, CA, May 1991.

[18] B. J. Falkowski and S. Rahardja, "Fast transforms for orthogonal logic," in *Proc. 1st IEEE Asia South Pacific Design Automation Conf.*, Chiba, Japan, 1995, pp. 385–390.

[19] D. Coppersmith, E. Feige, and E. Linzer, "Hadamard transform on multiply/add architectures," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Processing*, 1994, pp. 289–292.

[20] E. D. Dougherty and S. S. Agian, "Compression via orthogonal transform, wavelets, fractal, and logical systems," in *Short Course Notes, SPIE Conf. on AeroSense*, Orlando, FL, 1996.

[21] A. Drygajlo, "Butterfly orthogonal structure for fast transforms, filter banks and wavelets," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 1, pp. 59–71, 1992.

[22] R. C. Gonzalez and R. E. Woods, *Digital Image Processing*. Reading, MA: Addison-Wesley, 1993.

[23] B. J. Falkowski and M. A. Perkowski, "Algorithm and architecture for gray code ordered fast Walsh transforms," in *Proc. IEEE Int. Conf. Computer Communications*, Scottsdale, AZ, Mar. 1990, pp. 1596–1599.

[24] B. J. Falkowski, "Properties and ways of calculation of multi-polarity generalized Walsh transforms," *IEEE Trans. Circuits Syst.*, vol. 41, pp. 380–391, June 1994.

[25] B. J. Falkowski and C. H. Chang, "Generation of multi-polarity arithmetic transform from reduced representation of Boolean functions," in *Proc. 36th IEEE Midwest Symp. Circuits Systems*, Detroit, MI, 1995, pp. 2168–2171.

[26] P. Brodatz, *Textures—A Photographic Album for Artists and Designers* Dover, NY, 1966.

[27] A. K. Jain, *Fundamentals of Digital Signal Processing*. Englewood Cliffs, NJ: Prentice-Hall, 1989.

[28] R. O. Duda and P. E. Hart, *Pattern Classification and Scene Analysis*. New York: Wiley, 1973.

[29] K. Fukunaga, *Introduction to Statistical Pattern Recognition*. San Diego, CA: Academic, 1990.

[30] J. You and H. A. Cohen, "Classification and segmentation of rotated and scaled textured images using texture tuned masks," *Pattern Recognit.*, vol. 26, pp. 245–258, 1993.

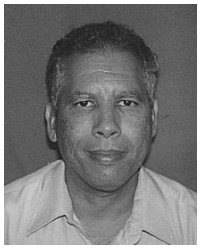
[31] J. G. Daugman, "Complete discrete 2-D Gabor transforms by neural networks for image analysis and compression," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. 36, pp. 1169–1179, 1988.

[32] K. Fukunaga K and R. R. Hayes, "Effect of sample size in classifier design," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 2, pp. 873–885, Aug. 1989.

[33] M. D. Swanson, "A binary wavelet decomposition of binary images," *IEEE Trans. Image Processing*, vol. 5, pp. 1637–1650, Dec. 1996.

Vidya Manian received the B.S. degree in electrical engineering from A.C. College of Engineering and Technology, Karaikudi, India, in 1990, and the M.S. degree from the University of Puerto Rico, Mayagüez, in 1995.

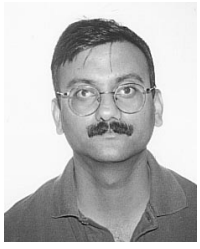
She is currently with the Department of Electrical and Computer Engineering, University of Puerto Rico, is the Processing Facility Manager for the Synthetic Aperture Radar Ground Station, Space Information Laboratory, and is also working in image processing at the Laboratory for Applied Remote Sensing and Image Processing. Her research interests include image classification, pattern recognition, parallel processing, and remote sensing.



Ramón Vásquez (S'73–M'77–SM'96) received the B.S. and M.S. degrees in electrical engineering from the University of Puerto Rico, Mayagüez, and the Ph.D. degree in electrical engineering from Louisiana State University, Baton Rouge, in 1984.

He is currently a Professor of electrical and computer engineering and the Dean of Engineering at the University of Puerto Rico. He is also the Director of the Center for Computational Research and Development, Department of Electrical and Computer Engineering. His research interests include computer vision, image processing, remote sensing, artificial intelligence, VLSI, and computer systems programming. He has chaired several sessions in educational and scientific conferences, and is one of the peer reviewers for various programs of NSF and NASA.

Dr. Vásquez is a Member of the PRS, SPIE, AASEE, ASPRS, ACM, Tau Beta Pi, Eta Kappu Nu, and Sigma Xi. He is listed in the *Who's Who in American Universities*.



Praveen Katiyar received the B.S. degree in science from Lucknow University, India, in 1985, the B.S. degree in electronics engineering from Chandrapur Engineering College, Nagpur, India, in 1989, and the M.S. degree in electrical engineering from the University of Puerto Rico, Mayagüez, in 1997.

He is currently a System Engineer with Telcordia Technologies. His research interests include image processing, texture analysis, and pattern recognition.