Support Vector Machines for Texture Classification

Kwang In Kim, Keechul Jung, Se Hyun Park, and Hang Joon Kim

Abstract—This paper investigates the application of support vector machines (SVMs) in texture classification. Instead of relying on an external feature extractor, the SVM receives the gray-level values of the raw pixels, as SVMs can generalize well even in high-dimensional spaces. Furthermore, it is shown that SVMs can incorporate conventional texture feature extraction methods within their own architecture, while also providing solutions to problems inherent in these methods. One-against-others decomposition is adopted to apply binary SVMs to multitexture classification, plus a neural network is used as an arbitrator to make final classifications from several one-against-others SVM outputs. Experimental results demonstrate the effectiveness of SVMs in texture classification.

Index Terms—Support vector machines, texture analysis, pattern classification, machine learning, feature extraction.

1 Introduction

Texture classification is basically the problem of classifying pixels in an image according to their textural cues. This is different from conventional image segmentation as the texture is characterized using both the gray value for a given pixel and the gray-level pattern in the neighborhood surrounding the pixel. Crucial to the success of texture classification are 1) the identification of features that differentiate textures in an image and developing their representations for further classification and 2) the construction of classification paradigms that operate on the above representations and discriminate between texture features associated with different texture classes. Accordingly, the texture classification problem is conventionally divided into the two subproblems of feature extraction and classification [1], [2], [3].

Many methods have been developed to extract textural features, which can loosely be classified as statistical, model-based, and signal processing methods. In statistical approaches, textures are described using statistical measures, such as the co-occurrence or autocorrelation statistics of the gray levels for k-tuples of pixels [4], [5]. The major drawback to this type of method is the enormous amount of data involved in kth order statistics, which is especially hard to handle when k is large ($k > 2$). This is relevant as psychophysical experiments have demonstrated that the human visual system is able to extract some statistics of an order higher than two [6]. Model-based methods characterize texture images based on probability distributions in random fields, such as Markov chains and Markov random fields (MRFs) [7], [9], [20], [30]. MRFs are widely used because they yield a local and economical texture description [7]. However, they also require intensive computation to determine the proper parameters [8]. Signal processing methods, also known as multichannel filtering methods, are attractive due to their simplicity [9]. In these methods, a textured input image is decomposed into feature images using a bank of filters, such as Gabor, wavelet, or neural network-based filters [10], [11], [12]. As a result, a high-dimensional textural pattern can be represented by a relatively small set of feature statistics that need to be extracted using a set of well-selected filters. Therefore, the major issue for this type of method is the selection of a good set of filters for a given texture classification problem.

From Bayes classifiers to neural networks, there are many possible choices for an appropriate classifier. Among these, support vector machines (SVMs) would appear to be a good candidate because of their ability to generalize in high-dimensional spaces, such as spaces spanned by texture patterns. The appeal of SVMs is based on their strong connection to the underlying statistical learning theory. That is, an SVM is an approximate implementation of the *structural risk minimization* (SRM) method [13]. For several pattern classification applications, SVMs have already been shown to provide better generalization performance than traditional techniques, such as neural networks [14], [15].

The aim of this paper is to illustrate the potential of SVMs in texture classification. Accordingly, a method for texture classification using SVMs is described. Unlike other texture classification methods, the proposed method does not externally incorporate any of the abovementioned feature extraction methods. Instead, the gray-level values of the raw pixels are directly fed to the SVM. The only preprocessing of the input before feeding it to the SVM is the selection of certain pixels following the configuration of autoregressive (AR) features. As a result, both feature extraction and classification are performed within the SVM architecture. The proposed method is based on the observation that an SVM incorporates feature extractors, therefore, nonlinear mapped input patterns can be used as feature vectors. Actually, in an SVM, feature extraction is implicitly performed by a kernel, which is defined as the dot product of two mapped patterns. It is also shown that one kernel (called a polynomial kernel) can perform the same operations as conventional feature extraction methods (specifically, statistical feature extraction and multichannel filtering). Furthermore, SVMs with such a kernel can provide solutions to the problems inherent in conventional feature extraction methods. The main advantage of the proposed method is that there is no need for a carefully designed feature extraction mechanism because the feature extraction task is reduced to the problem of training the SVMs. Thereafter, feature extraction and classification are both performed in accordance with a unique criterion referred to as the *classification rate*.

Since SVMs were originally developed for two-class problems, their basic scheme is extended to multitexture classification by adopting the *one-against-others* decomposition method. This works by applying SVMs that first separate one class from all the other classes, and then arbitrating between several SVMs. A neural network is used as the arbitrator and the results are compared with the commonly used *max-selection* scheme. The proposed method was tested using several Brodatz and MIT Vision Texture images. The excellent classification rates achieved in the experiments confirm that SVMs are well-suited for texture classification.

2 Overview of SVMs

An SVM constructs a binary classifier from a set of labeled patterns called *training examples*. Let $(\mathbf{x}_i, y_i) \in \mathbf{R}^N \times \{\pm 1\}, i = 1, \ldots, l$ be such a set of training examples. The purpose is to select the function $f_\alpha : \mathbf{R}^N \rightarrow \{\pm 1\}$ from a given class of functions $\{f_\alpha : \alpha \in \Lambda\}$ such that f will correctly classify test examples $(\mathbf{x}, y)$. If no restriction is placed on the class of functions when choosing the estimate f, even a function that performs well with training data may not generalize well to unseen examples. Hence, just minimizing the training error

- *K.I. Kim is with the Artificial Intelligence Lab, Computer Science Department, Korea Advanced Institute of Science and Technology, Taejon, 305-701, Korea. E-mail: kimki@ai.kaist.ac.kr.*
- *K. Jung is with the Pattern Recognition and Image Processing Lab, Computer Science and Engineering Department, 3115 Engineering Bldg., Michigan State University, MI 48824-1226. E-mail: jungke@msu.edu.*
- *S.H. Park is with the Computer Engineering Division, College of Electronics and Information, Chosun University, Gwangju, 501-759, Korea. E-mail: sehyun@chosun.ac.kr.*
- *H.J. Kim is with the Computer Engineering Department, Kyungpook National University, Taegu, 702-701, Korea. E-mail: hjkim@ailab.knu.ac.kr.*

Manuscript received 31 May 2000; revised 14 June 2001; accepted 21 Feb. 2002.
Recommended for acceptance by A. Kundu.
For information on obtaining reprints of this article, please send e-mail to: tpami@computer.org, and reference IEEECS Log Number 112204.

TABLE 1
Possible Kernel Functions and Types of Classifier

Inner product kernel	Type of classifier
Polynomial kernel: $K(\mathbf{x},\mathbf{y}) = (\mathbf{x}\cdot\mathbf{y})^p$	Polynomial learning machine
Gaussian kernel: $K(\mathbf{x},\mathbf{y}) = \exp(-\frac{1}{2\sigma^2}\|\mathbf{x}-\mathbf{y}\|^2)$	Radial-basis function network
Tangent hyperbolic kernel: $K(\mathbf{x},\mathbf{y}) = \tanh(\mathbf{x}\cdot\mathbf{y}-\Theta)$	Two-layer perceptron

does not imply a small test error averaged over test examples drawn from an underlying distribution $P(\mathbf{x}, y)$.

Statistical learning theory [13] shows that it is imperative to restrict the class of functions so that f is chosen from a class with a *capacity* that is suitable for the amount of available training data. As such, this theory provides bounds for the test error. The minimization of these bounds, which depend on both the empirical risk and the capacity of the function class, leads to the principle of *structural risk minimization* (SRM) [13], [16]. The best-known capacity concept of this theory is the VC-dimension, defined as the largest number of points that can be separated in all possible ways using the functions of the given class [13].

To design learning algorithms, the capacity of the class of functions selected needs to be computed. The SVM selects the class of separating hyperplanes whose VC-dimension bounds can be computed and then identifies the optimal separating hyperplane (OSH) that maximizes the margin of the nearest examples. This is equivalent to minimizing the VC-dimension bound. The OSH is then computed as a decision surface of the form:

$$f(\mathbf{x}) = \operatorname{sgn}\left(\sum_{i=1}^{l^s} y_i \alpha_i \mathbf{x}_i^s \cdot \mathbf{x} + b\right), \tag{1}$$

where $\{\mathbf{x}_i^s\}_{i=1}^{l^s}$ is a subset of the training data set. These subsets are called *support vectors* (SVs) and are the points from the data set that fall closest to the separating hyperplane. The coefficients α_i and b are determined by solving the large-scale quadratic programming problem:

Maximize

$$W(\alpha) = \sum_{i=1}^{l} \alpha_i - \frac{1}{2}\sum_{i,j=1}^{l} \alpha_i \alpha_j y_i y_j (\mathbf{x}_i \cdot \mathbf{x}_j)$$

subject to the constraints:

$$\sum_{i=1}^{l} \alpha_i y_i = 0, \quad 0 \le \alpha_i \le C \quad \text{for } i = 1, \ldots, l.$$

The parameter C can be regarded as the regularization parameter and is selected by the user. A larger C corresponds to assigning a higher penalty to the training errors.

However, since it is unlikely that a general pattern recognition problem can actually be solved by a linear classifier (separating hyperplane), the OSH needs to be augmented in order to allow for nonlinear decision surfaces. The basic idea is to map the data into another dot product space (called the *feature space*) F via a nonlinear map $\Phi : \mathbf{R}^N \to F$ and perform the above linear algorithm in F. Since the solution has the form

$$f(\mathbf{x}) = \operatorname{sgn}\left(\sum_{i=1}^{l^s} y_i \alpha_i \Phi(\mathbf{x})_i^s \cdot \Phi(\mathbf{x}) + b\right), \tag{2}$$

it is nonlinear in the original input variables.

The mapping Φ is performed in accordance with Cover's theorem on the seperability of patterns [17]. Consider a space made up of nonlinearly separable patterns. Cover's theorem states that such a multidimensional space can be transformed into a new feature space F where the patterns are linearly separable with a high probability, provided two conditions are satisfied: First, the transformation is nonlinear and second, the dimensionality of the feature space is high enough. Accordingly, F usually needs to have a very high dimensionality in order to be linearly separable. This introduces the important problem of how to treat such high-dimensional space computationally. This can be resolved based on two observations: First, although some mappings have very high dimensionalities, their inner products can be easily computed, and second, all the Φ mappings used in the SVM occur in the form of an inner product. Accordingly, the solution is to replace all the occurrences of an inner product resulting from two mappings with the kernel function K defined as:

$$K(\mathbf{x}, \mathbf{y}) = \Phi(\mathbf{x}) \cdot \Phi(\mathbf{y}).$$

Conversly, given a symmetric positive kernel $K(\mathbf{x}, \mathbf{y})$, Mercer's theorem [29] implies the existance of a mapping Φ such that $K(\mathbf{x}, \mathbf{y}) = \Phi(\mathbf{x}) \cdot \Phi(\mathbf{y})$. Then, without considering the mapping Φ explicitly, a nonlinear SVM can be constructed by selecting the proper kernel. Table 1 summarizes the kernel functions for three common types of SVMs: polynomial learning machines, radial basis function networks (RBFNs), and two-layer perceptrons [13], [16].

3 Texture Classification Using Support Vector Machines

This section describes the classification system devised to assess the potential of SVMs in texture classification. First, some useful properties of SVMs for texture classification are discussed in Section 3.1. Based on these properties, Section 3.2 outlines the representation of texture patterns from an input image and introduces an SVM classifier for a two-class texture classification problem. Section 3.3 describes the SVM-neural network architecture for more general multiclass texture classification problems. Finally, the postprocessing method is presented in Section 3.4.

3.1 Optimality of SVMs for Texture Classification

The operation of an SVM for texture classification is two-fold:

1. The nonlinear mapping of a texture space into a possibly high-dimensional feature space.
2. The construction of an OSH in the feature space.

Step 2 is explained in Section 2. Therefore, the current section discusses the optimality of Step 1 for texture representations i.e., texture feature extraction, from two different perspectives:

- **Statistical feature extraction.** By introducing various kernel functions (for example, see Table 1), various Φ mappings can be used, which are implicitly imbedded in the SVMs. One of these mappings (induced by a polynomial kernel) takes the p-order correlations between the entries x_i of the input vector $\mathbf{x}$. If $\mathbf{x}$ represents a texture pattern with entires that are pixel values, this amounts to mapping the input space into the space of the pth order products (monomials) of the input pixels. It should be noted that the direct computation of this type of feature is not easy, even for a moderate sized problem, as N-dimensional input patterns include $N_F = (N+p-1)!/\,p!(N-1)!$ different monomials,

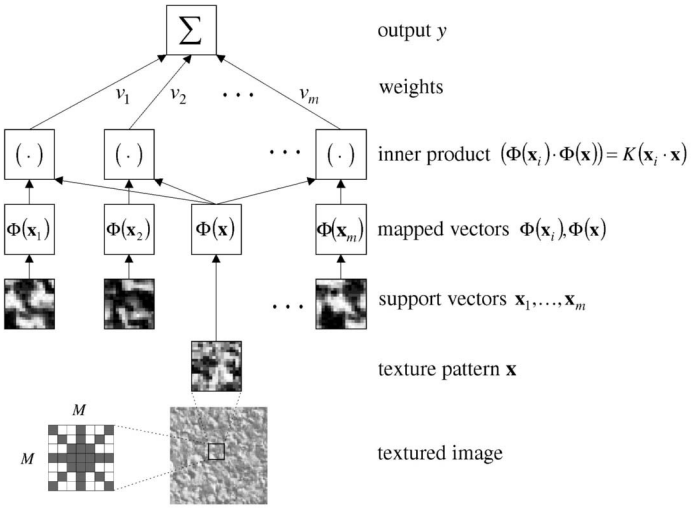

Fig. 1. Architecture of two-class SVM texture classifier.

comprising a feature space F of dimensionality N_F. As such, a 17×17 texture pattern and monomial degree $p = 5$ yield a feature space dimensionality of about 10^8. However, introducing a polynomial kernel facilitates work in such a space, as a polynomial kernel with degree p corresponds to the dot product of the feature vectors extracted by the monomial feature extractor C_p [13]:

$$\begin{aligned}\big(C_p(\mathbf{x})\cdot C_p(\mathbf{y})\big) &= \sum_{i_1,\ldots,i_p=1}^{N} x_{i_1}\cdot\ldots\cdot x_{i_p}\cdot y_{i_1}\cdot\ldots\cdot y_{i_p}\\ &= \left(\sum_{i=1}^{N} x_i y_i\right)^p = (\mathbf{x}\cdot\mathbf{y})^p.\end{aligned}$$

- **Signal processing.** In multichannel filtering methods, a textured input image is decomposed into a set of feature images using a bank of filters. The decomposition is accomplished by filtering the input image, applying a nonlinear function to all the pixels in the filtered images, and spatially smoothing each output image [10]. The channels corresponding to different filters are assumed to capture certain specific local characteristics of the input texture, such as the spatial frequency, directionality, edgeness, etc. Yet, the major issue in multichannel filtering is the filter selection problem: That is, the selection of a moderate number of filters that can capture the textural properties of the given classification problem. This filter selection problem has been tackled both empirically [18] and theoretically [19], [20]. However, the practical implementation of both types of solution, tend to be computationally prohibitive, or result in a suboptimal solution. The operation performed by a kernel in an SVM is essentially the same as that of a channel in a multichannel filtering method: The kernel first computes the inner product $\mathbf{z} = \mathbf{x}\cdot\mathbf{y}$ between its input $\mathbf{x}$ and the SVs $\mathbf{y}$ and then performs a nonlinear transformation Θ. In the case of a polynomial kernel, this transformation corresponds to a polynomial function $(\Theta(\mathbf{z}) = (\mathbf{z})^p)$. The only difference is that the spatial smoothing of the feature values is omitted, as in the proposed method, this is performed in a postprocessing stage. In the new method, the SVs play the role of a filter bank. Even though they are not designed as filters for capturing specific frequency bands or orientations, they can still extract critical measures for classification. As such, it should be noted that an SVM can emphasize the correct separation of training data through the selection of filters (SVs), while, in a classical multichannel filtering approach, the filter selection criteria are generally not directly related to classification performance. In this respect, an SVM texture classifier is rather similar to a neural network-based multichannel filtering method [12]. However, in contrast to a neural network, an SVM is more concerned with minimizing the test error than with directly minimizing the training error. Futhermore, in an SVM, the number of filters and their coefficients are both automatically determined by the number of SVs and their values, respectively. By selecting the SVs as special-purpose filters optimized for given textures, the SVM can automatically identify an optimal feature extractor and the corresponding texture classifier based on a unified criterion of minimizing the test error for a given texture classification task.

3.2 Data Representation and Classification of Two-Class Textures

The simplest way to characterize the variability in a texture pattern is by noting the gray-level values of the raw pixels. This set of gray values becomes the feature set on which the classification is based. An important advantage of this approach is the speed with which images can be processed, as the features do not need to be calculated. However, the main disadvantage is that the size of the feature vector is large. Accordingly, a texture classifier may need to generalize texture patterns in a high-dimensional space. SVMs are the proper tool for this problem because they are capable of learning in sparse and high-dimensional spaces with very few training examples. Furthermore, SVMs themselves incorporate feature extractors and can use their nonlinear mapped input

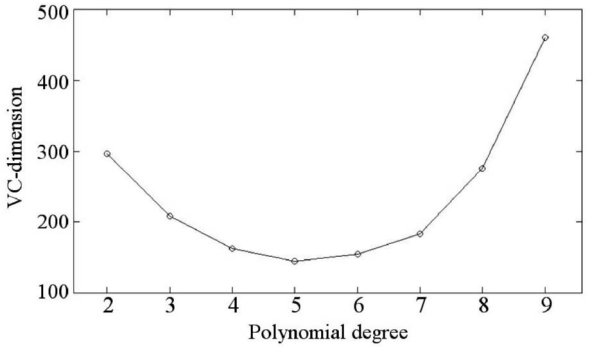

Fig. 2. Estimated VC-dimension of SVMs for polynomial degrees 2 through 9.

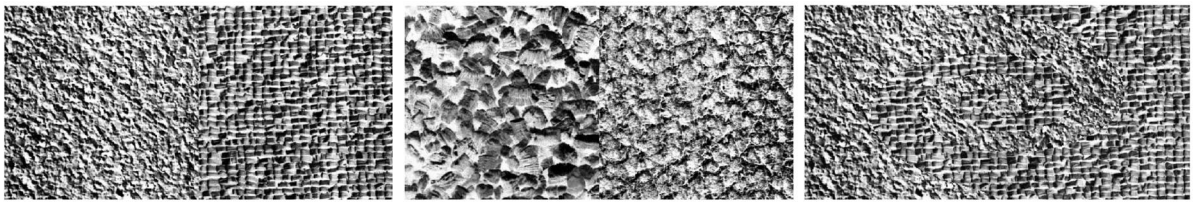

Fig. 3. Two-texture images used in experiments.

pattern $\Phi(\mathbf{x})$ as features for classification (see Section 3.1). Therefore, it may be advantageous to allow SVMs to extract features directly from the pixels rather than forcing them to base their features on a different user-defined feature set.

Fig. 1 shows the architecture of an SVM texture classifier, which involves three layers with entirely different roles. The input layer is made up of source nodes that connect the SVM to its environment. Its activation $\mathbf{x}$ comes directly from the gray-level values of the $M \times M$ window in the input image. However, instead of using all the pixels in the window, the input configuration for extracting AR features (only the shaded pixels from the input window in Fig. 1) [9] is used. This reduces the size of the feature vector from M^2 to $4M$-3, thereby resulting in an improved generalization performance and classification speed. The hidden layer applies a nonlinear transformation from the input space to the feature space F, where the inner products are computed. These two operations are in practice performed in a single step using the kernel function. The size of the hidden layer m is determined as the number of SVs identified by the SVM. It should also be noted that the SVs help shape and better define the decision surface, when compared with other patterns and are border patterns from a geometrical point of view. Consequently, since there are usually very few SVs, the size of the hidden layer is moderate. The output layer is a hyperplane classifier, supplying the response of the network to the activation pattern applied to the input layer. The sign of the SVM output y represents the class of the central pixel in the input window.

The parameters that need to be tuned in the SVMs include the kernel parameters and regularization parameter C. One straightforward method for tuning is to use validation data, which are distinct from the training data. However, when the amount of available training data is limited, it is important to have an alternative means of tuning the parameters, without having to put aside parts of the training set for validation purposes. The proposed method for tuning the parameters is mainly based on Schölkopf et. al's work [21].

The strength of an SVM is based in its automatic capacity control (in terms of the VC-dimension). This capacity control takes place within a class of functions $\{f_\alpha : \alpha \in \Lambda\}$ specified a priori by the choice of the kernel function (or equivalently F). Schölkopf et. al pointed out that the VC-dimension of f_α can be estimated by a term that is in proportion to the radius r of the smallest ball containing all the data points in F [21]. Since r depends on the shape of F, the VC-dimension depends on the kernel parameters (p, in the case of a polynomial kernel). The estimation of r from the given kernel parameters can be formulated in a similar way to training SVMs [21]. The VC-dimension and training errors can then be compared to select the optimal kernel parameters. The proposed strategy is to select those parameters that minimize r, while retaining a zero training error in the trained system in the corresponding F. In the case of a polynomial kernel, the condition for a zero training error would appear to be reasonable because preliminary experiments on the classification of several texture images have shown that the training error diminishes when $p \geq 4$. However, for more complex problems, this condition could be relaxed. For the same reason, parameter C does not have a significant effect on the classification because, in the case of a zero training error, it is not included in the computation of the OSH. Accordingly, a C value of 100 was used in the current study for training the SVMs. Readers interested in the automatic selection of C are referred to [22].

TABLE 2
Training Errors for Fig. 3a as a Function of p

p	1	2	3	4	5	6-9
Error rate (%)	28.1	7.5	2.1	0	0	0

A two-class texture classification problem was considered as an example (Fig. 3a). SVMs with a varying p were trained with 2,000 training examples obtained using random selections of 1,000 patterns from both classes. The input window size was fixed at 17×17. The estimated VC-dimensions and training errors for different degrees are shown in Fig. 2 and Table 2. From the results, the optimal degree for this problem was determined to be 5, which exhibited the minimal VC-dimension and produced zero training errors.

3.3 Multiclass Texture Classification

The previous section described an SVM for a two-class texture classification problem, called a *dichotomy*. This may also be appropriate for texture-based object detection applications when discriminating an object from the background. However, the majority of texture classification problems involve more than two textures. Consequently, extended SVMs are required for application to multiple textures, and the optimal design of multiclass SVM classifiers is still an area of active research. One frequently used method is *one-against-others* decomposition, which works by constructing an SVM ω_r for each class r that first separates that class from all the other classes and then uses an expert F to arbitrate between each SVM output in order to produce the final decision. The effectiveness of this method in the problem of texture classification has already been demonstrated using HMM-based classifiers [23].

A *max-selector* is the simplest form of arbitrator. If $\mathbf{g} = (g^1, \ldots, g^R)^T$ denotes the output[1] of a system of R one-against-others SVMs, the *max-selector* picks class r for the input $\mathbf{x}$, which then maximizes g^r as defined by

$$F = \arg\max_r(\mathbf{g}).$$

However, a max-selector suffers from a scaling problem because it assumes that all the gs are on the same scale, which is not the case for SVMs. If the SVM is trained to produce outputs for the SVs as ± 1, the scale is not robust as it only depends on a few data, often including outliers [24]. In a max-selector, the output class is determined by choosing the maximum of all the SVM outputs. However, the outputs of the remaining SVMs, other than the winner, also carry certain information. Moreover, the mean of g can vary significantly according to the class of input [24]. This

1. The distance from the OSH obtained by removing $\mathrm{sgn}(\cdot)$ from (1) and (2).

(a) (b) (c)

Fig. 4. Multitexture images used in experiments.

TABLE 3
Sources of Test Images

Image	Sources
Figs. 3(a) and 3(c)	D4 and D84 from [25]
Fig. 3(b)	D5 and D92 from [25]
Fig. 4(a)	D4, D9, D19, and D57 from [25]
Fig. 4(b)	Fabric.0007, Fabric.0009, Leaves.0003, Misc.0002, and Sand.0000 from [26]
Fig. 4(c)	Fabric.0000, Fabric.0007, Flowers.0005, Food.0005, Grass.0001, Metal.0002, Misc.0002, Sand.0000, and Stone.0004 from [26]

knowledge can be used to improve the overall recognition performance. In [24], a stacking technique based on linear mapping was applied for normalizing $\mathbf{g}$. While this technique shows significant improvements over the bare *one-against-others* decomposition, preliminary experiments have indicated that a linear normalization is often insufficient for texture classification as the relation among gs becomes nonlinear. In the current work, after uniformly scaling $\mathbf{g}$ by applying a tangent hyperbolic function $\mathbf{h} = (h, \ldots, h)^T$, a nonlinear mapping $M : \mathbf{R}^R \rightarrow \mathbf{R}^R$ is used to aggregate the answers of all the SVMs into a score for each class. Thus, the arbitrator can be defined by

$$F = \arg\max_r(M(\mathbf{h}(\mathbf{g}))).$$

A two-layer neural network, composed of a size 2 hidden layer with a tangent hyperbolic activation function, is adopted for the mapping M. The network is designed to minimize the mean square error between $\mathbf{h}(\mathbf{g}(\mathbf{x}))$ and the desired output $\mathbf{y} = (-1, \ldots, +1, \ldots -1)^T$ and trained using a back-propagation algorithm.

3.4 Postprocessing

Any method can result in a classification error. In the proposed method, misclassified pixels usually take the form of noisy speckles scattered across the entire classification image. In this case, simple image smoothing techniques can reduce the error rate by an order of magnitude. In the current study, two-dimensional median filtering with a window size of five was used for postprocessing the classified image. The application of this method improved the error rate by an average of 7 percent. However, it is worth noting that the error rate created by a smoothing technique may depend on the size and shape of the uniformly textured regions.

4 Experimental Results

To verify the effectiveness of the proposed method, experiments were performed on a supervised segmentation of several test images. The test images were drawn from two different commonly used texture sources: the Brodatz album [25] and MIT Vision Texture (VisTex) database [26] (See Figs. 3 and 4). Table 3 summarizes the sources of the test images.

All textures were originally gray-scale images with 256 levels. The texture classifier was trained on randomly selected portions of 256×256 subimages of texture images that were not included in the test images. To make the textures indiscriminable for the local mean gray level or local variance, both the training and test images were globally histogram-equalized before being used and the gray scales linearly normalized into $[-1, 1]$. The desired classes for the patterns were then manually assigned. Patterns lying on the texture boundaries were excluded from the training set due to the difficulties involved in manually assigning the desired value for the training pattern. Accordingly, this introduced a tradeoff when choosing the appropriate window size for the classification accuracy of uniform texture regions and texture boundaries, as the shape and size of the texture boundaries and input window size all affect the performance of the classification method. Therefore, the experiments were performed with different input window sizes of 5×5, 9×9, 13×13, 17×17, and 21×21.

Polynomial kernels were emphasized because of their relevance to conventional feature extraction methods, yet the performance of the SVMs with other kernels, including a linear SVM (with a polynomial kernel of degree 1), was also tested. All the experiments were performed using a 500 MHz Pentium3 CPU. The time spent classifying an image depended on the number of texture classes (equal to the number of SVMs), input window size, type of kernels, and number of SVs. It took about one minute to classify the entire image of Fig. 3a using a degree-5 polynomial kernel and 17×17 window.

4.1 Two-Texture Images

The sizes of the images in Fig. 3 are 512×256 and the size of each textured subregion is half that of the test image. The sources of the texture classes in Fig. 3c are the same as those in Fig. 3a. However, since the texture boundary in Fig. 3c is more complex and larger than that in Fig. 3a, a larger window size was considered to be disadvantageous for classifying Fig. 3c, when compared with Fig. 3a. The training data set was obtained by randomly selecting

TABLE 4
Error Rates and Number of SVs for Figs. 3a and 3b with a 17×17 Window Using Varying Polynomial Degrees

Polynomial degree p		1	3	5	7	9
Fig. 3(a)	Error rate	29.4%	11.6%	8.6%	8.5%	9.1%
	# of SVs	698	142	135	149	146
Fig. 3(b)	Error rate	31.3%	14.1%	11.9%	12.0%	11.4%
	# of SVs	854	309	332	342	327

TABLE 5
Error Rates (%) with Different Window Sizes

Kernel	Image	Window size				
		5×5	9×9	13×13	17×17	21×21
Polynomial kernel (p=5)	Fig. 3(a)	17.8	12.7	9.4	8.6	13.0
	Fig. 3(b)	20.5	14.6	12.1	11.9	15.6
	Fig. 3(c)	20.2	15.4	12.2	13.3	17.9
Gaussian kernel (σ=0.5)	Fig. 3(a)	17.3	13.1	11.8	10.6	13.2
	Fig. 3(b)	21.3	15.2	13.2	12.3	17.2
	Fig. 3(c)	21.6	17.1	12.9	13.7	18.5
Tangent hyperbolic kernel (Θ=1.5)	Fig. 3(a)	20.4	16.4	15.5	13.1	15.3
	Fig. 3(b)	24.3	19.7	15.3	14.9	19.2
	Fig. 3(c)	28.1	21.4	16.3	17.9	21.5

1,000 patterns from each texture. This corresponded to about 1.7 percent of the total available input patterns.

To evaluate the effect of the polynomial degree p on the texture classification performance, a set of experiments was performed with varying degrees of p. The input window size was fixed at 17×17. Table 4 shows the error rates and numbers of SVs for the classification of Figs. 3a and 3b. The highest recognition rates were obtained using degree-7 and -9 polynomial kernels. However, no obvious optimum was identified. Higher degrees of p afforded a low error rate, while a saturation point was reached when $p \geq 5$. Although not optimal in testing, a degree-5 polynomial kernel for Fig. 3a, estimated from the training set (see Section 3.2), was identified as second best, which was only slightly behind the best (0.1 percent). For a degree-1 polynomial kernel (i.e., linear SVM), the problem was not separable and the error rate was rather high. In this case, all the training errors were related to the SVs, thereby causing a larger number of SVs. The number of SVs only slightly increased with an increasing polynomial degree. When compared with the number of training patterns, the relatively large number of SVs revealed that the texture-learning task for Fig. 3b was rather difficult to solve. Since no significant difference was observed across the classification results when $p \geq 5$, a fixed value of five was used for p in the rest of the experiments.

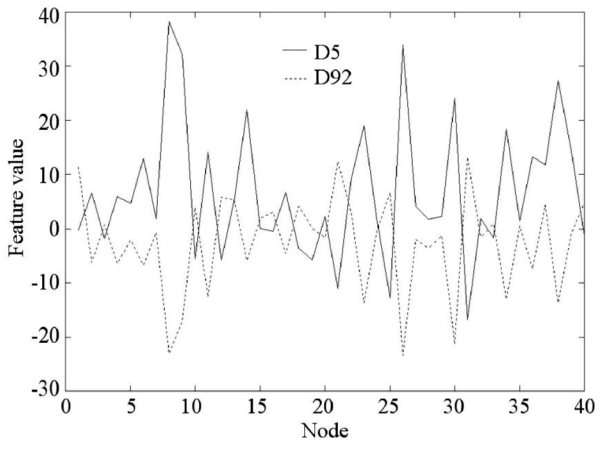

Fig. 5. First 40 components of average feature vectors for textures D5 and D92.

The error rates according to the various window sizes are summarized in Table 5. The tendency for a smaller error rate was observed with larger input window sizes, except for a 21×21 window. The 17×17 window exhibited the best overall performance for the images in Figs. 3a and 3b. For more complex texture boundaries (Fig. 3c), the unstable classification of those patterns lying on the texture boundaries resulted in a larger error rate for a 17×17 window. For comparison, classification results using Gaussian kernels and tangent hyperbolic kernels are also presented, where the parameters ($\sigma = 0.5$ for Gaussian kernels and $\Theta = 1.5$ for tangent hyperbolic kernels) were empirically selected. The highest recognition rates were obtained when using a polynomial kernel. The Gaussian kernels were slightly outperformed by the polynomial kernels, whereas the tangent hyperbolic kernels produced much higher error rates.

It should be noted that Fig. 3b was very difficult to discriminate for several texture feature extraction approaches, including Laws filters (a bank of band pass filters [28]), optimal representation Gabor filters, etc. [9], whereas an SVM with a polynomial kernel resulted in a rather low error rate. This demonstrates the usefulness of a monomial feature extractor induced by a polynomial kernel. Since it is not easy to visualize the feature space when the polynomial degree is larger than 2, the operation of this space was indirectly observed through the kernel responses (activation of hidden layer in Fig. 1). Fig. 5 shows the first 40 components of the average kernel responses calculated from each texture class (D5, D92). The distinct *signatures* observed from each texture class highlight the role of the hidden layer as a discriminative feature extractor for a given texture classification problem. It should also be noted that the operation of the hidden layer is essentially the same as that of a multichannel filtering scheme (see Section 3.1).

TABLE 6
Error Rates (%) with Postprocessing

Window size	Fig. 3(a)	Fig. 3(b)	Fig. 3(c)
13×13	1.1	0.8	2.3
17×17	0.3	0.2	2.9

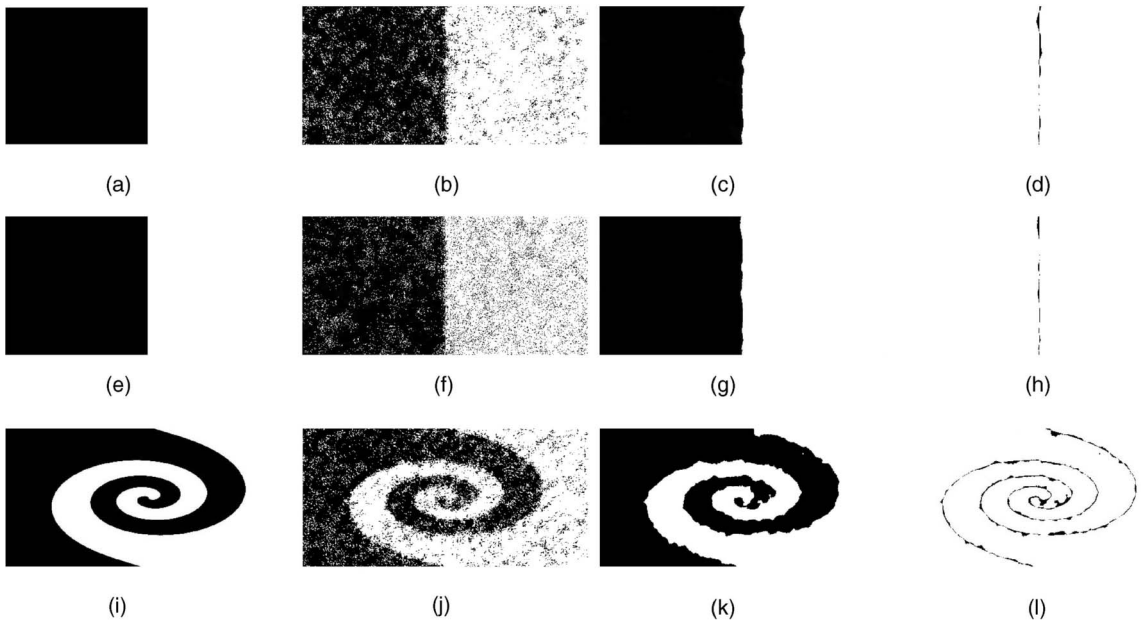

Fig. 6. Classification results for two texture classes: (a), (e), and (i) reference images, (b), (f), and (j) classification results for Figs. 3a, 3b, and 3c using only texture classifier, (c), (g), and (k) postprocessed results of (b), (f), and (j), respectively, and (d), (h), and (l) misclassified pixels illustrated as black-gray levels.

TABLE 7
Error Rates (%) for Multitexture Images

Window size	5×5	9×9	13×13	17×17	21×21
Fig. 4(a)	28.8	22.3	17.3	16.1	21.8
Fig. 4(b)	28.5	21.8	20.0	18.5	19.7
Fig. 4(c)	29.9	27.7	22.9	24.2	29.9

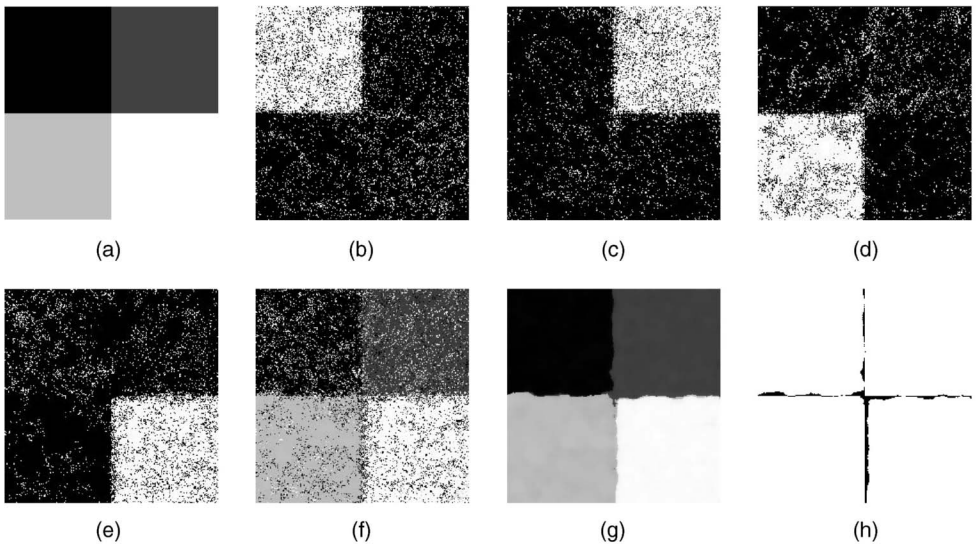

Fig. 7. Classification results for four texture classes: (a) reference image, (b), (c), (d), (e) intermediate results obtained from classification using individual SVMs, (f) classification results using SVM and neural network, (g) postprocessed result, and (h) misclassified pixels illustrated as black-gray levels.

The error rates in Table 5, except for Fig. 3b, were higher than those reported in previous texture analysis literature because no smoothing or averaging of the outputs was used. Accordingly, when the classification results were postprocessed, the error rates were substantially reduced (Table 6 and Fig. 6). In particular, in the case of Fig. 3b, postprocessing resulted in almost perfect classification. However, since this technique means the error rates depend heavily on the size and shape of uniform texture regions, the analysis of the classification results in the current study concentrated on those results obtained without postprocessing.

4.2 Multitexture Images

Fig. 4a shows a 256×256-size test image composed of four Brodatz textures. 2,000 training patterns (500 from each texture class) were used to train four SVMs. The best error rate obtained was 16.1 percent when using a 17×17 window (Table 7), while the average error rate of each one-against-others was 9.9 percent. It

TABLE 8
Classification Results Using Various Multiclass Extension Methods

Method	Error rate (%)
One-against-others	21.2
One-against-others with linear normalization [24]	17.2
One-against-others with neural network	15.9
One-against-one	19.9
All-at-once [27]	22.4
Average	19.3

should be noted that the classification errors occurring in the individual SVMs did not necessarily end up as errors in the arbitrated classification image. Fig. 7 shows the classification results.

To understand the relevance of the results obtained with the proposed multiclass extension method for SVMs, a comparison was made of the classification results obtained from several different extension methods, including simple one-against-others, one-against-others with linear normalization [24], one-against-one, and all-at-once. For the all-at-once method, the implementation of a library for support vector machines (LIBSVM) made by Hsu and Lin [27] was utilized. Table 8 summarizes the results. Scaling with a neural network significantly improved one-against-others, which ranked as the best, meanwhile linear normalization and one-against-one came in second and third, respectively, and all-at-once showed a rather inferior performance.

Fig. 4b shows a 256×256-size test image composed of 5 VisTex textures. To train the classifier, 500 patterns were randomly selected from each texture class. The best error rate was 18.5 percent when using a 17×17 window (Table 7). Fig. 4c shows another textured image consisting of nine VisTex textures. The image size is 384×384 and the size of each textured subregion is 128×128. A total of 3,600 patterns were randomly selected (400 patterns from each texture class) as the training set, corresponding to about 2.4 percent of the total input patterns. The best error rate was 22.9 percent, obtained when using a 13×13 window. In Fig. 4b, most of the errors were related to classifying the first texture class (Fabric.0007) due to difficulties in discriminating between the first and third texture classes (Fabric.0007 and Leaves.0003). In contrast, in Fig. 4c, many of the misclassified pixels were near the boundaries between the different textures (more than 50 percent of the total errors identified). Fig. 8 shows the classification results of the two images.

4.3 Comparisons

So far, SVMs have only been applied directly to the gray values of raw pixels. This configuation is based on two observations: SVMs work well even in high-dimensional spaces, plus they incorporate feature extractors within their own architecture. Yet, there is still a question as to the quality of the classification result when external feature extraction is used with an SVM. Accordingly, experiments were performed using one of the most commonly used feature extraction methods for texture classification. A multichannel filtering method was used based on Gabor filters with four different orientations ($0°$, $45°$, $90°$, and $135°$) and four high-radial frequencies ($64/\sqrt{2}$, $32/\sqrt{2}$, $16/\sqrt{2}$, and $8/\sqrt{2}$ cycles per image), as described in [10]. The classification of the extracted features was performed using SVMs with a polynomial kernel of degree 5. To make the comparison fair, the feature images were not averaged before being fed to the classifier. In addition, for benchmark comparisons with other classification methods, experiments were also performed using neural networks. The networks included two hidden layers of size 20 with tangent hyperbolic activation functions and were trained using a back-propagation algorithm minimizing the mean squared error. Following the studies of Jain and Karu [12], where neural networks were also used to classify textures, the number of training steps was fixed at 1,000,000. To avoid local minima, the reported results were obtained by training 10 neural networks with different initial weights, and then selecting the minimal error over all the results.

Table 9 shows the results for the classification of Figs. 3a and 3b. For comparison, the results obtained from the SVM are also presented. With Gabor filters, a slight improvement was observed in Fig. 3a, while a significant increment in the error rate was observed in Fig. 3b. As such, the ability to achieve acceptable results without the use of Gabor filters indicates the possibility of using SVMs directly on texture patterns. In contrast, the better performance of the SVM over the neural network suggests that the decision boundary generated by the SVM was more effective in generalizing the given set of texture patterns.

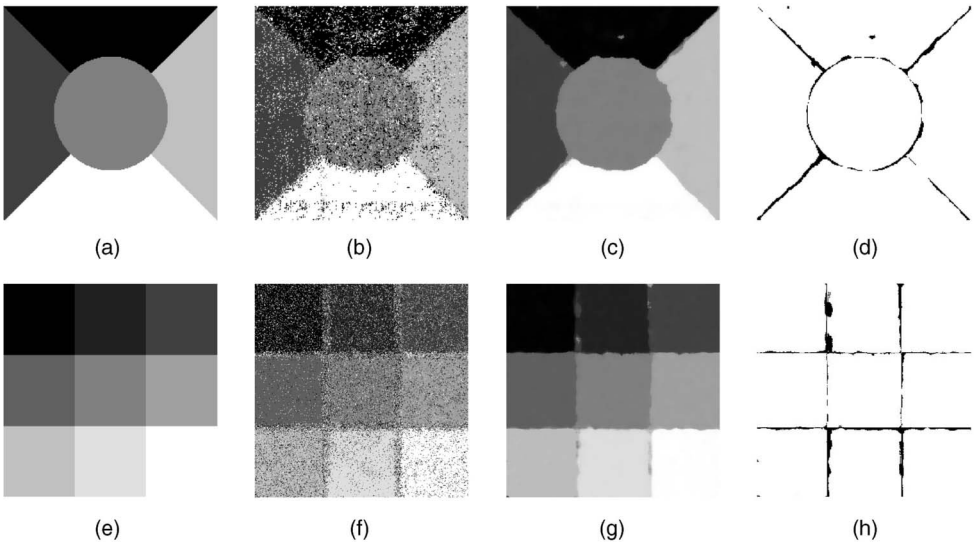

Fig. 8. Texture classification results for Figs. 4b and 4c: (a) and (e) reference images, (b) and (f) classification results, (c) and (g) classification results after postprocessing, and (d) and (h) misclassified pixels illustrated as black-gray levels.

TABLE 9
Error Rates (%) Using Different Classifiers

	Neural networks	SVMs	Gabor filters + SVMs
Fig. 3(a)	13.9	8.6	8.3
Fig. 3(b)	17.8	11.9	16.4

5 Conclusions

This paper described an SVM-based texture classification method to assess the potential of SVMs in texture classification. With the exception of adopting the input configuration from the AR features, the proposed method does not use any external feature extraction scheme. Instead, the SVMs receive the gray-level values of the raw pixels as the input pattern. The rationale for this configuration is that an SVM has the capability of learning in high-dimensional spaces, such as gray-level texture pattern spaces, plus it can incorporate a feature extraction scheme within its own architecture. The validity of using polynomial kernels for texture feature extraction was also discussed from two feature extraction perspectives. The excellent performance achieved by an SVM when using these kernels also supported these views. For a multitexture classification problem, one-against-others decomposition is adopted and a neural network used to arbitrate the outputs of each SVM.

Future studies will investigate incorporating more texture-specific knowledge within SVM architectures. Possible choices include the selection of a feature space induced by specific kernels. For example, a Fourier-domain analysis would be helpful in extracting specific frequency and orientation components. Some of the main applications of this method include page-layout segmentation, text location in a video image, and textured object detection. Accordingly, further experiments are required related to real-world applications along with the fine-tuning of certain parameters, such as the input window size and structure of the arbitrator.

References

[1] H. Greenspan, R. Godman, R. Chellappa, and C.H. Anderson, "Learning Texture-Discrimination Rules in a Multiresolution System," *IEEE Trans. Pattern Analysis Machine Intelligence,* vol. 16, no. 9, pp. 894-901, Sept. 1994.

[2] A.K. Muhamad and F. Deravi, "Neural Networks for the Classification of Image Texture," *Eng. Applications of Artificial Intelligence,* vol. 7, pp. 381-393, 1994.

[3] Y.Q. Chen, M.S. Nixon, and D.W. Thomas, "On Texture Classification," *Int'l J. Systems Science,* vol. 28, no. 7, pp. 669-682, 1997.

[4] M. Tuceryan and A.K. Jain, "Texture Analysis," *Handbook Pattern Recognition and Computer Vision,* C.H. Chen, L.F. Pau, and P.S.P. Wang, eds., Singapore: World Scientific, pp. 235-276, 1993.

[5] R. Haralick, K. Shangmugam, and L. Dinstein, "Textural Features for Image Classification," *IEEE Trans. Systems, Man, and Cybernetics,* vol. 3, pp. 610-621, 1973.

[6] P. Diaconis and D. Freedman, "On the Statistics of Vision: The Julesz Conjecture," *J. Math. Psychology,* vol. 24, 1981.

[7] S.Z. Li, *Markov Random Field Modeling in Computer Vision.* New York: Springer-Verlag, 1995.

[8] H.J. Kim, E.Y. Kim, J.W. Kim, and S.H. Park, "MRF Model Based Image Segmentation Using Hierarchical Distributed Genetic Algorithm," *IEE Electronics Letters,* vol. 34, no. 25, pp. 1394-1395, 1998.

[9] T. Randen and J.H. Husoy, "Filtering for Texture Classification: A Comparative Study," *IEEE Trans. Pattern Analysis Machine Intelligence,* vol. 21, no. 4, pp. 291-310, 1999.

[10] A.K. Jain and F. Farrokhnia, "Unsupervised Texture Segmentation Using Gabor Filters," *Pattern Recognition,* vol. 24, no. 12, pp. 1167-1186, 1991.

[11] C.S. Lu, P.C. Chung, and C.F. Chen, "Unsupervised Texture Segmentation via Wavelet Transform," *Pattern Recognition,* vol. 30, no. 5, pp. 729-742, 1997.

[12] A.K. Jain and K. Karu, "Learning Texture Discrimination Masks," *IEEE Trans. Pattern Analysis Machine Intelligence,* vol. 18, no. 2, pp. 195-205, Feb. 1996.

[13] V. Vapnik, *The Nature of Statistical Learning Theory.* New York: Springer-Verlag, 1995.

[14] B. Schölkopf, K. Sung, C.J.C. Burges, F. Girosi, P. Niyogi, T. Pogio, and V. Vapnik, "Comparing Support Vector Machines with Gaussian Kernels to Radial Basis Function Classifiers," *IEEE Trans. Signal Processing,* vol. 45, no. 11, pp. 2758-2765, 1997.

[15] S. Haykin, *Neural Network—A Comprehensive Foundation,* second ed. Prentice Hall, 1999.

[16] C.J.C. Burges, "A Tutorial on Support Vector Machines for Pattern Recognition," *Data Mining and Knowledge Discovery,* vol. 2, no. 2, pp. 1-47, 1998.

[17] T.M. Cover, "Geometrical and Statistical Properties of Systems of Linear Inequalities with Applications in Pattern Recognition," *IEEE Trans. Electronic Computers,* vol. 14, pp. 326-334, 1965.

[18] K.I. Laws, "Textured Image Segmentation," PhD thesis, Univ. of Southern Calif., 1980.

[19] D.F. Dunn, W.E. Higgins, and J. Wakeley, "Determining Gabor Filter Parameters for Texture Segmentation," *Proc. SPIE Intelligence Robots and Computer Vision XI,* pp. 51-63, 1992.

[20] S.C. Zhu, Y. Wu, and D. Mumford, "Filters, Random Fields and Maximum Entropy (FRAME)—Towards a Unified Theory for Texture Modeling," *Int'l J. Computer Vision,* vol. 27, no. 2, 1998.

[21] B. Schölkopf, C. Burges, and V. Vapnik, "Extracting Support Data for a Given Task," *Proc. Int'l Conf. Knowledge Discovery & Data Mining,* pp. 252-257, 1995.

[22] B. Schölkopf, "Support Vector Learning," PhD thesis, Munich: Oldenbourg Verlag, 1997.

[23] J.-L. Chen and A. Kundu, "Unsupervised Texture Segmentation Using Multichannel Decomposition and Hidden Markov Models," *IEEE Trans. Image Processing,* vol. 4, no. 5, pp. 603-619, 1995.

[24] E. Mayoraz and E. Alpaydin, "Support Vector Machines for Multi-Class Classification," Technical Report IDIAP-PR 98-06, Dalle Molle Inst. for Perceptual Artificial Intelligence, 1998.

[25] P. Brodatz, *Textures: A Photographic Album for Artists and Designers.* New York: Dover, 1966.

[26] *MIT Vision and Modeling Group.* 1998.

[27] C.-W. Hsu and C.-J. Lin, "A Comparison on Methods for Multi-Class Support Vector Machines," technical report, Dept. of Computer Science and Information Eng., Nat'l Taiwan Univ., 2001.

[28] K.I. Laws, "Rapid Texture Identification," *Proc. SPIE Conf. Image Processing for Missile Guidance,* pp. 376-380, 1980.

[29] J. Mercer, "Functions of Positive and Negative Type, and Their Connection with the Theory of Integral Equations," *Trans. London Philosophical Soc. (A),* vol. 209, pp. 415-446, 1909.

[30] A.L. Yuille, J. Coughlan, S.C. Zhu, and Y. Wu, "Order Parameters for Minimax Entropy Distributions: When Does High Level Knowledge Help?" *Proc. IEEE Conf. Computer Vision and Pattern Recognition,* pp. 558-565, 2000.