Texture Classification using the Support Vector Machines

Shutao Li [a,b] James T. Kwok [a] Hailong Zhu [a,c] Yaonan Wang [b]

[a] *Department of Computer Science, Hong Kong University of Science and Technology, Clear Water Bay, Hong Kong*

[b] *College of Electrical and Information Engineering, Hunan University, Changsha, People's Republic of China*

[c] *GE Global Research, GE China Technology Center Co., Ltd., Shanghai, People's Republic of China*

Abstract

In recent years, support vector machines (SVMs) have demonstrated excellent performance in a variety of pattern recognition problems. In this paper, we apply SVMs for texture classification, using translation-invariant features generated from the discrete wavelet frame transform. To alleviate the problem of selecting the right kernel parameter in the SVM, we use a fusion scheme based on multiple SVMs, each with a different setting of the kernel parameter. Compared to the traditional Bayes classifier and the learning vector quantization algorithm, SVMs, and, in particular, the fused output from multiple SVMs, produce more accurate classification results on the Brodatz texture album.

Key words: Texture classification; Support vector machines; Discrete wavelet frame transform
Corresponding author: James Kwok (Fax: +852-2358-1477, Email: jamesk@cs.ust.hk)

1 Introduction

With the increasing popularity of digital libraries and multimedia databases, content-based image retrieval is becoming an important research topic. A very useful characterization for a wide range of images is by their textures. Moreover, it is generally believed that even our human visual system uses textures for recognition and interpretation. In recent decades, textures have been successfully used for object recognition, scene interpretation and segmentation, especially in the areas of industrial automation, remote sensing and medical diagnosis. Numerous techniques have also been developed for texture classification, segmentation, synthesis and other related tasks.

An important step in texture classification is feature extraction. In the past few decades, a large number of texture features have been proposed. Tuceryan and Jain [1] categorized these into four major methods, namely, statistical (e.g., the co-occurrence method [2]), geometrical (e.g., the Voronoi tessellation features [3]), model-based (e.g., the Markov random fields [4], fractals [5] and the multiresolution autoregressive features [6]) and signal processing (e.g., the Gabor filters [7], wavelet transforms [8,9] and quadrature mirror filters [10]). A few empirical comparisons on these texture features have been conducted, though sometimes with contradicting results [11–13]. Recently, Randen and and Husϕy [14] performed a more systematic and comprehensive study, with particular emphasis on the signal processing approach. They concluded that no texture feature is consistently superior for all images, though the wavelet frame and quadrature mirror filters are among the best for most images.

Besides having good empirical performance, wavelets, in particular, have solid foundations based on a formal mathematical theory for multiresolution image analysis [9]. However, one problem with the commonly used discrete wavelet transform (DWT) is that it is not translation-invariant. A simple integer shift of the signal/image will result in non-trivial modifications of the wavelet coefficients. This is problematic in texture analysis as textures are generally considered as translation-invariant. Unser [15] alleviated this problem by using an overcomplete wavelet representation, namely the discrete wavelet frame transform (DWFT). It avoids the down-sampling operations in DWT and thus yields a description that is translation-invariant.

Another crucial component in texture classification and segmentation is the classifier being used. Again, a number of classifiers have been used, including the nearest neighbor classifiers [16], Fisher linear discriminant [17], neural networks [8], learning vector quantization [14], and, being the commonest of all, the Bayes classifier [18,19,8,20,21,15]. In recent years, the support vector machine (SVM) [22,23] has emerged as a very successful pattern recognition method. It has outperformed traditional techniques in various applications such as handwritten digit recognition [24], text categorization [25–27], spam categorization [28] and 3D object recognition [29]. Besides, SVMs have been extended and also yielded superior performance in other problems such as regression [30,31] and time series prediction [32]. An important advantage of the SVM is that it is based on the principle of structural risk minimization [22], rather than on empirical risk minimization as do many other methods. Thus, SVMs aim at minimizing a bound on the generalization error, instead of simply minimizing the training error. Moreover, on the computational aspect, SVM training can be reduced to a quadratic programming problem. Consequently, the solution found is always globally optimal. Besides, unlike other pattern recognition methods, SVM does not depend explicitly on the dimensionality of the problem, but to the "margin" with which it separates the data. Thus, superior performance can be achieved even on high-dimensional data sets. For example, it outperforms radial basis networks on recognizing the US postal service database of handwritten digits (using 16×16 bitmap as input) [24]. Promising results are also reported in a face detection experiment [33], where the input dimensionality is 283.

In this paper, we apply SVMs to the problem of texture classification, with translation-invariant texture features generated by DWFT. Recently, SVMs have also been applied to the related tasks of content-based image retrieval with relevance feedback [34] and for learning image similarity [35]. The rest of this paper is organized as follows. Brief introductions to both DWFT and SVM are given in Sections 2 and 3 respectively. Section 4 describes the texture classification method and procedures used in the experiments, and the last section gives some concluding remarks.

2 Discrete Wavelet Frame Transform

The wavelet transform [9,36] represents any arbitrary function as a superposition of wavelets, which are functions generated from a mother wavelet ψ by dilations and translations, as

$$\psi^{a,b}(x) = \frac{1}{\sqrt{a}}\psi(\frac{x-b}{a}).$$

In image processing, we are mostly interested in the two-dimensional extension of the wavelet transform. Figure 1 shows one stage of the 2-D DWT and inverse DWT. Here, DWT is implemented using a bank of 1-D low-pass (h) and high-pass (g) analysis filters, and the inverse DWT is likewise computed via 1-D synthesis filters, $\tilde{h}$ and $\tilde{g}$. After one stage of decomposition, an image at resolution level i will be decomposed into four subband images $D_{LH}^{i+1}, D_{HL}^{i+1}, D_{HH}^{i+1}$ and S_{LL}^{i+1}. The three details images, $D_{LH}^{i+1}, D_{HL}^{i+1}$ and D_{HH}^{i+1}, correspond to the low-high, high-low and high-high bands in the frequency domain respectively. The remaining (low-low) component, S_{LL}^{i+1}, is a low-pass filtered version of S_{LL}^{i}, and can be used for further subband decomposition. A DWT with d decomposition levels will thus result in a total of $3d+1$ subbands. Moreover, notice that in Figure 1 we have a biorthogonal wavelet basis, i.e., the analysis filters h and g are different from the synthesis filters $\tilde{h}$ and $\tilde{g}$. Biorthogonal wavelet bases are usually more desirable than orthogonal ones because they can preserve linear phase, have finite impulse response, and the mother wavelets can have arbitrarily high regularity [37].

DWFT resembles its DWT counterpart except that the dyadic down-sampling process is not imposed at each level (Figure 2). Thus, unlike DWT, DWFT yields a shift invariant signal representation. As with DWT, performing DWFT to d decomposition levels also results in a total of $3d$ details subbands and one approximation subband. However, while the sizes of the subbands in DWT decrease as the decomposition goes on, each subband in DWFT has the same size as the original image (Figure 3). This redundant representation of DWFT is thus more demanding in terms of both memory and time.

3 Support Vector Machines

In this Section, we will briefly review the use of SVMs in binary classification problems. Interested readers may consult [38–40,22,23] for details and other applications.

Let the training set D be $\{(\mathbf{x}_i, y_i)\}_{i=1}^N$, with each input $\mathbf{x}_i \in \Re^m$ and the output label $y_i \in \{\pm 1\}$. The SVM first maps $\mathbf{x}$ to $\mathbf{z} = \phi(\mathbf{x})$ in a Hilbert space $\mathcal{F}$ (with inner product $\langle \cdot, \cdot \rangle$) via a nonlinear map $\phi : \Re^m \to \mathcal{F}$. This space $\mathcal{F}$ is often called the *feature space*, and its dimensionality is usually very high (sometimes infinite). Consider the case when the data is linearly separable in $\mathcal{F}$, i.e., there exists a vector $\mathbf{w} \in \mathcal{F}$ and a scalar $b \in \Re$ such that

$$y_i(\langle \mathbf{w}, \phi(\mathbf{x}_i) \rangle + b) \geq 1,$$

for all elements in the training set. The SVM constructs a hyperplane $(\langle \mathbf{w}, \phi(\mathbf{x}) \rangle + b)$ for which the separation between the positive and negative examples is maximized (Figure 4). It can be shown [41] that the $\mathbf{w}$ for this "optimal" hyperplane can be found by minimizing $\|\mathbf{w}\|$, and the resultant solution can be written as $\mathbf{w} = \sum_{i=1}^N \alpha_i y_i \phi(\mathbf{x}_i)$ for some $\alpha_i \geq 0$. The vector of α_i's, $\mathbf{\Lambda} = (\alpha_i, \ldots, \alpha_N)$, can be found by solving the following quadratic programming (QP) problem [1]:

$$\text{maximize } W(\mathbf{\Lambda}) = \mathbf{\Lambda}^T \mathbf{1} - \frac{1}{2} \mathbf{\Lambda}^T \mathbf{Q} \mathbf{\Lambda}, \tag{1}$$

with respect to $\mathbf{\Lambda}$, subject to the constraints $\mathbf{\Lambda} \geq \mathbf{0}$ and $\mathbf{\Lambda}^T \mathbf{Y} = 0$. Here, $\mathbf{Y}^T = (y_1, \ldots, y_N)$ and $\mathbf{Q}$ is a symmetric matrix with elements

$$Q_{ij} = y_i y_j \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) \rangle. \tag{2}$$

[1] Here, superscript T denotes the vector transpose operation.

For those α_i's greater than zero, the corresponding training examples must lie along the margins of the decision boundary (by the Kuhn-Tucker theorem), and these are called the *support vectors*.

However, recall that the dimensionality of $\mathcal{F}$, and thus also of $\phi(\mathbf{x}_i)$ and $\phi(\mathbf{x}_j)$ in (2), are usually very high. Hence, in order for this approach to be practical, a key characteristic of the SVM, and of kernel methods in general, is that one can obtain $\langle\phi(\mathbf{x}_i), \phi(\mathbf{x}_j)\rangle$ in (2) without having to obtain $\phi(\mathbf{x}_i)$ and $\phi(\mathbf{x}_j)$ explicitly first. This is achieved by using a *kernel* function $K(\cdot,\cdot)$ such that

$$K(\mathbf{x}_i, \mathbf{x}_j) = \langle\phi(\mathbf{x}_i), \phi(\mathbf{x}_j)\rangle. \tag{3}$$

For example, it can be shown that the kernel

$$K(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i^T \mathbf{x}_j + 1)^d \tag{4}$$

corresponds to a map ϕ into the space spanned by all products of exactly d dimensions of $\Re^m$ [23]. More generally, it can be shown that any function satisfying Mercer's theorem can be used as kernel, and each such kernel will have an associated map ϕ such that (3) holds [23]. Another commonly used function that also satisfies the Mercer conditions is the Gaussian kernel

$$K(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2}), \tag{5}$$

where σ controls its width. Here, d in (4) and σ in (5) are usually called the kernel parameters.

Computationally, SVMs (and kernel methods in general) also have the important advantage that only quadratic programming[2], and not nonlinear optimization, is involved. As $\mathbf{Q}$ is always positive semi-definite, there is no local maximum while maximizing (1). The use of kernels thus provides an elegant nonlinear generalization of many existing linear algorithms [39,44].

During testing, for a test vector $\mathbf{x} \in \Re^m$, we first compute the activation

$$s = \langle\mathbf{w}, \phi(\mathbf{x})\rangle + b = \sum_{i=1}^{N} \alpha_i y_i K(\mathbf{x}, \mathbf{x}_i) + b.$$

With a user-defined threshold s_0, the class label for $\mathbf{x}$ is then assigned by the

[2] Recently, the SVM problem has also been formulated as a linear programming problem [42,43] instead of a quadratic programming problem.

following rule:

$$\text{label} = \begin{cases} 1 & s > s_0, \\ -1 & \text{otherwise.} \end{cases}$$

When the training set is not separable in $\mathcal{F}$, the SVM algorithm tries to minimize $\|\mathbf{w}\|$ while at the same time separating the data with a minimum number of errors. This is done by minimizing

$$\frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{N}\xi_i, \tag{6}$$

with $\xi_i \geq 0$ satisfying the constraint:

$$y_i(\langle \mathbf{w}, \phi(\mathbf{x}_i)\rangle + b) \geq 1 - \xi_i.$$

Here, C is a regularization parameter controlling the tradeoff between model complexity (the first term in (6)) and training error (the second term) in order to ensure good generalization performance. Again, minimization of (6) can be transformed to a QP problem: maximize (1) under the constraints $\mathbf{0} \leq \mathbf{\Lambda} \leq C\mathbf{1}$ and $\mathbf{\Lambda}^T\mathbf{Y} = 0$.

4 Method and Procedures

In this paper, we use SVMs for texture classification, with the texture features generated by the translation-invariant DWFT using the $coif1$ wavelet basis. The original texture image is first decomposed using DWFT. Denote the $M \times N$ image obtained in subband i by I_i. Based on the common belief that a texture can be identified by its energy distribution in the frequency domain, the local energies

$$\frac{1}{M \times N}\sum_{x=1}^{M}\sum_{y=1}^{N} I_i^2(x, y)$$

from each filtered subband image (except the one from the approximation subband) are then used as texture features. By decomposing the image to d levels, we thus obtain a feature vector of length $3d$.

Typically, the number of texture classes is greater than two. Thus, unlike the binary classification problem discussed in Section 3, we have a multi-class classification problem. Here, we adopt the conventional approach of casting this as a number of binary classification problems. In other words, we have one classifier for each texture class, each attempting to separate samples belonging to this class from samples belonging to the other classes. On classifying a new sample, the classifier with the largest output will be selected as the winner, and this new sample is assigned to the winner's corresponding texture class.

In the experiments, the Gaussian kernel in (5) will be used in the SVM. Preliminary results suggest that the Gaussian kernel outperforms the polynomial kernel in (4). However, one still has to determine a proper value for the kernel parameter σ in (5). Typically, this has to be tuned manually. A more disciplined approach is to use a validation set [32], or by data-resampling techniques such as cross-validation and bootstrapping. However, these methods can be very expensive in terms of computation time and/or training data. Alternatively, one can utilize an upper bound on the generalization error predicted by the theory of SRM [22]. Experiments in [38,22,45] indicated that the bounds are very loose, though the minimum of the bound seems to approximately coincide with the minimum of the generalization error. More recently, a number of Bayesian methods [46–48] have also been devised to tackle this problem, though they can still be quite computationally expensive.

In this paper, we use instead a simple fusion scheme based on voting among multiple SVMs, each with a different setting of σ. These different SVMs are trained using the same training set. Upon the arrival of a new sample, it will be assigned to the texture class predicted by the majority of these SVMs. If such a majority solution cannot be found, we will then assign to the new sample the texture class that corresponds to the SVM with maximum output value. In this experiment, three SVMs are used, with σ equals to 0.1, 1 and 5 respectively. The computational requirement is thus roughly three times the case when just one SVM is used. For simplicity, we always use the default value of 1000 for the parameter C in (6), though in principle one can also extend this fusion scheme to multiple SVMs with different values of C.

We compare SVM with the commonly used Bayes classifier for minimum error rate classification [49]. The feature vector in texture class i is assumed to follow the Gaussian distribution $N(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)$, where $\boldsymbol{\mu}_i$ and $\boldsymbol{\Sigma}_i$ are the corresponding mean and covariance matrix for class i. On classifying a test pattern $\mathbf{x}$, the Bayes classifier first computes a distance (discriminant) function between $\mathbf{x}$ and each texture class, then $\mathbf{x}$ is assigned to the class with minimum distance. The most commonly used distance functions are:

(1) the Euclidean distance $\|\mathbf{x} - \boldsymbol{\mu}_i\|^2$,
(2) the Bayes distance $(\mathbf{x} - \boldsymbol{\mu}_i)^T \boldsymbol{\Sigma}_i^{-1} (\mathbf{x} - \boldsymbol{\mu}_i) + \log |\boldsymbol{\Sigma}_i|$,
(3) the Mahalanobis distance $(\mathbf{x} - \boldsymbol{\mu}_i)^T \boldsymbol{\Sigma}_i^{-1} (\mathbf{x} - \boldsymbol{\mu}_i)$, and

(4) the simplified Mahalanobis distance: $\sum_{j=1}^{J}(x_j - \mu_{ij})^2/\sigma_{ij}$, where $\boldsymbol{\mu}_i = (\mu_{i1}, \mu_{i2}, \ldots, \mu_{iJ})^T$, $\mathbf{x} = (x_1, x_2, \ldots, x_J)^T$, σ_{ij} is the jth diagonal element of $\boldsymbol{\Sigma}_i$ and J is the dimensionality of the feature vector.

In the following, we will only report results on using the Bayes distance and the Mahalanobis distance, which have better performance in previous texture classification experiments [18] as well as in our preliminary study. Besides, we also compare with the learning vector quantization (LVQ) algorithm [50], which has been commonly used in texture classification. In the experiment, we use the optimized-learning-rate LVQ1 algorithm (with default parameter settings) from the LVQ_PAK package (Version 3.1) [3]. Moreover, all the features are normalized to the range 0–1.

4.1 Data

In this experiment, we select a set of 30 natural textures (Figure 5) from the Brodatz album [51]. Each texture image is of size 640×640 with 256 gray levels. Each image is subdivided into 400 non-overlapping 32×32 subimages [4]. As noted in [14], many previous texture classification studies used overlapping training and test sets, and this is likely to yield unreliable and over-optimistic performance results. Here, we use a complete separation between the training and test sets. A small fraction (from 1.25% to 10%) of the 400 subimages are used in training the classifiers, while the rest are used for testing. Moreover, to further randomize the experiment, 10 random partitions of the training and test sets are used. The classification accuracy, averaged over all 30 texture classes and also over these 10 random partitions, will be used to evaluate the results.

4.2 Experimental Results

Performance of the various classifiers using different numbers of training samples, with 3, 4 and 5 levels of wavelet decomposition, are shown in Tables 1, 2 and 3 respectively (and also graphically in Figure 6). All classifiers attain reasonably good results, even though only a small proportion of the subimages, each covering only a small part of the whole image, are used for training. The superiority of the fused SVM over both Bayes classifiers (with the Bayes distance and the Mahalanobis distance), LVQ and the individual SVMs can

[3] The LVQ_PAK package can be downloaded from http://www.cis.hut.fi/research/lvq_pak/.

[4] In selecting the images from the Brodatz album, we have tried to ensure that their visual properties are similar over the whole image.

also be clearly seen. Moreover, notice that when the number of decomposition levels increases from 4 to 5, the performance of the Bayes classifiers and LVQ do not improve much or in some cases even deteriorate. This is in line with previous observations (e.g., in [14]) on using traditional pattern recognition techniques that increasing the number of texture features may not be beneficial. In contrast, there is a more marked improvement when SVMs are used. This, again, demonstrates that SVMs are less affected by the curse of dimensionality and are able to glean useful information from the additional features.

To compare the performance of the classifiers in more detail, Table 4 and Figure 7 show the individual classification accuracy for each texture class (when 5 decomposition levels and 10% of the samples are used for training). As can be seen, in general, when the textures can be easily classified, all classifiers have comparable performance. On the other hand, when the task of texture classification becomes hard, the fused SVM has much better performance than the Bayes classifiers and LVQ.

5 Conclusion

In this paper, we apply SVMs to the task of texture classification, using translation-invariant texture features generated by DWFT as input. To alleviate the problem of selecting a proper value for the kernel parameter in SVM training, we use a fusion scheme based on simple voting among multiple SVMs, each with a different setting of the kernel parameter. Experiments are performed on a subset of natural textures from the Brodatz album. Compared to the traditional Bayes classifier and LVQ, SVMs, in general, produce more accurate classification results. Moreover, the voting combination of multiple SVMs is always more accurate than any one of the individual SVMs (unless the training data set is very small).

Here in the experiment, relatively simple subband energies obtained from DWFT have been used as texture features. In the future, we plan to use more complicated texture features, including using wavelet transforms that are not only translation-invariant, but are also rotation- and scale-invariant (e.g., [20]), with applications to content-based image retrieval. Moreover, the voting scheme used in Section 4 is just one possible way to combine multiple classifiers in general. Other possibilities such as weighted voting, bagging and mixture of experts [52] will also be investigated in the future.

Acknowledgments

This research has been partially supported by the Research Grants Council of the Hong Kong Special Administrative Region under grants HKUST2033/00E and HKUST6195/02E. The authors would like to thank Gavin Cawley for his Matlab SVM toolbox [53].

References

[1] M. Tuceryan and A.K. Jain. Texture analysis. In C.H. Chen, L.F. Pau, and P.S.P. Wang, editors, *Handbook of Pattern Recognition and Computer Vision*, pages 235–276. World Scientific, 1993.

[2] R. Haralick, K. Shanmugam, and I. Dinstein. Textural features for image classification. *IEEE Transactions on Systems, Man, and Cybernetics*, 3:610–621, November 1973.

[3] M. Tuceryan and A.K. Jain. Texture segmentation using Voronoi polygons. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12:211–216, 1990.

[4] R. Chellappa and S. Chatterjee. Classification of textures using Gaussian Markov random fields. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 33:959–963, 1985.

[5] J.M. Keller, S. Chen, and R.M. Crownover. Texture description and segmentation through fractal geometry. *Computer Vision and Graphical Image Processing*, 45:150–166, 1989.

[6] J. Mao and A.K. Jain. Texture classification and segmentation using multiresolution simultaneous autoregressive models. *Pattern Recognition*, 25(2):173–188, 1992.

[7] A.K. Jain and F. Farrokhnia. Unsupervised texture segmentation using Gabor filters. *Pattern Recognition*, 24(12):1167–1186, 1991.

[8] A. Laine and J. Fan. Texture classification by wavelet packet signatures. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(11):1186–1191, November 1993.

[9] S.G. Mallat. A theory for multiresolution signal decomposition: The wavelet representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(7):674–693, July 1989.

[10] T. Randen and J.H. Husϕy. Multichannel filtering for image texture segmentation. *Optical Engineering*, 33(8):2617–2625, August 1994.

[11] M. Pietikänien, A. Rosenfeld, and L.S. Davis. Experiments with texture classification using averages of local pattern matches. *IEEE Transactions on Systems, Man, and Cybernetics*, 13(3):421–426, 1983.

[12] J. Strand and T. Taxt. Local frequency features for texture classification. *Pattern Recognition*, 27(10):1397–1406, 1994.

[13] J.S. Weszka, C.R. Dyer, and A. Rosenfeld. A comparative study of texture measures for terrain classification. *IEEE Transactions on Systems, Man, and Cybernetics*, 6:269–285, April 1976.

[14] T. Randen and J.H. Husϕy. Filtering for texture classification: A comparative study. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(4):291–310, 1999.

[15] M. Unser. Texture classification and segmentation using wavelet frames. *IEEE Transactions on Image Processing*, 4(11):1549–1560, 1995.

[16] Y. Chitre and A.P. Dhawan. M-band wavelet discrimination of natural textures. *Pattern Recognition*, 32(5):773–789, 1999.

[17] D.A. Clausi and M.Ed Jernigan. Designing Gabor filters for optimal texture separability. *Pattern Recognition*, 33(11):1835–1849, 2000.

[18] T. Chang and C.-C.J. Kuo. Texture analysis and classification with tree-structured wavelet transform. *IEEE Transactions on Image Processing*, 2(4):429–441, 1993.

[19] N.-D. Kim and S. Udpa. Texture classification using rotated wavelet filters. *IEEE Transactions on Systems, Man, and Cybernetics, Part A: Systems and Humans*, 30(6):847–852, November 2000.

[20] V. Manian and R. Vasquez. Scaled and rotated texture classification using a class of basis functions. *Pattern Recognition*, 31(12):1937–1948, 1998.

[21] G. Menegaz, A. Rivoldini, and J-Ph. Thiran. Dyadic frames of directional wavelets as texture descriptors. *Wavelets Applications in Signal and Image Processing VIII*, 4119:263–273, 2000.

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

[23] V. Vapnik. *Statistical Learning Theory.* Wiley, New York, 1998.

[24] B. Schölkopf, K. Sung, C. Burges, F. Girosi, P. Niyogi, T. Poggio, and V. Vapnik. Comparing support vector machines with Gaussian kernels to radial basis function classifiers. *IEEE Transactions on Signal Processing*, 45(11):2758–2765, November 1996.

[25] S. Dumais. Using SVMs for text categorization. *IEEE Intelligent Systems*, 13(4):21–23, July/August 1998.

[26] T. Joachims. Text categorization with support vector machines: Learning with many relevant features. In *European Conference on Machine Learning*, 1998.

[27] J.T. Kwok. Automated text categorization using support vector machine. In *Proceedings of the International Conference on Neural Information Processing*, pages 347–351, Kitakyushu, Japan, October 1998.

[28] H. Drucker, D. Wu, and V.N. Vapnik. Support vector machines for spam categorization. *IEEE Transactions on Neural Networks*, 10(5):1048–1054, 1999.

[29] M. Pontil and A. Verri. Support vector machines for 3D object recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(6):637–646, 1998.

[30] A.J. Smola and B. Schölkopf. A tutorial on support vector regression. NeuroCOLT2 Technical Report NC2-TR-1998-030, Royal Holloway College, 1998.

[31] V. Vapnik, S.E. Golowich, and A. Smola. Support vector method for function approximation, regression estimation, and signal processing. In M.C. Mozer, M.I. Jordan, and T. Petsche, editors, *Advances in Neural Information Processing Systems 9*, pages 281–287. Morgan Kaufmann, San Mateo, CA, 1997.

[32] K.R. Müller, A.J. Smola, G. Rätsch, B. Schölkopf, J. Kohlmorgen, and V. Vapnik. Predicting time series with support vector machines. In *Proceedings of the International Conference on Artificial Neural Networks*, 1997.

[33] E. Osuna, R. Freund, and F. Girosi. Training support vector machines: an application to face detection. In *Proceedings of Computer Vision and Pattern Recognition*, Puerto Rico, June 1997.

[34] P. Hong, Q. Tian, and T.S. Huang. Incorporate support vector machines to content-based image retrieval with relevance feedback. In *Proceedings of the International Conference on Image Processing*, volume 3, pages 750–753, Vancouver, Canada, 2000.

[35] G.D. Guo, S.Z. Li, and K.L. Chan. Learning similarity for texture image retrieval. In *Proceedings of the European Conference on Computer Vision*, Dublin, Ireland, 2000.

[36] S.G. Mallat. *A Wavelet Tour of Signal Processing.* Academic Press, San Diego, 2nd edition, 1999.

[37] M. Antonini, M. Barlaud, P. Mathieu, and I. Daubechies. Image coding using wavelet transform. *IEEE Transactions on Image Processing*, 1(2):205–220, 1992.

[38] C.J.C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2):955–974, 1998.

[39] N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines.* Cambridge University Press, 2000.

[40] K.-R. Müller, S. Mika, G. Rätsch, K. Tsuda, and B. Schölkopf. An introduction to kernel-based learning algorithms. *IEEE Transactions on Neural Networks*, 12(2):181–202, 2001.

[41] C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 20:273–297, 1995.

[42] O.L. Mangasarian and D.R. Musicant. Robust linear and support vector regression. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(9), 2000.

[43] A. Smola, B. Schölkopf, and Rätsch G. Linear programs for automatic accuracy control in regression. In *Proceedings of the International Conference on Artificial Neural Networks*, 1999.

[44] B. Schölkopf, C. Burges, and A. Smola, editors. *Advances in Kernel Methods – Support Vector Learning.* MIT Press, Cambridge, MA, 1999.

[45] B. Schölkopf. *Support Vector Learning.* PhD thesis, Technische Universität Berlin, 1997.

[46] J.T. Kwok. The evidence framework applied to support vector machines. *IEEE Transactions on Neural Networks*, 11(5):1162–1173, 2000.

[47] M. Seeger. Bayesian model selection for support vector machines, Gaussian processes and other kernel classifiers. In S.A. Solla, T.K. Leen, and K.-R. Müller, editors, *Advances in Neural Information Processing Systems 12*, San Mateo, CA, 1999. Morgan Kaufmann.

[48] P. Sollich. Bayesian methods for support vector machines: Evidence and predictive class probabilities. *Machine Learning*, 46(1/3):21–52, 2002.

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

[50] T. Kohonen. *Self-Organizing Maps.* Springer, Berlin, third edition, 2001.

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

[52] E. Alpaydin. Techniques for combining multiple learners. In *Proceedings of Engineering of Intelligent Systems*, volume 2, pages 6–12, 1998.

[53] G.C. Cawley. MATLAB support vector machine toolbox, 2000. University of East Anglia, School of Information Systems, Norwich, Norfolk, U.K. NR4 7TJ.

[54] T.G. Dietterich. Approximate statistical tests for comparing supervised classification learning algorithms. *Neural Computation*, 10(7):1895–1924, 1998.

Vitae

SHUTAO LI was born in 1972 in P.R. China. He received the B.S., M.S., and Ph.D. degrees in electrical engineering from the Hunan University, Changsha, P.R. China in 1995, 1997, and 2001, respectively. He joined the College of Electrical and Information Engineering, Hunan University, in 2001. He was with Department of Computer Science, Hong Kong University of Science and Technology, from May 2002 to October 2002. He is currently a Research Staff Member with the Department of Computer Science, Royal Holloway College, University of London. His main research interests include image processing and machine learning.

JAMES T. KWOK received the Ph.D. degree in computer science from the Hong Kong University of Science and Technology in 1996. He then joined the Department of Computer Science, Hong Kong Baptist University as an Assistant Professor. He returned to the Hong Kong University of Science and Technology in 2000 and is now an Assistant Professor in the Department of Computer Science. His research interests include kernel methods, artificial neural networks, pattern recognition and machine learning.

HAILONG ZHU was born in 1973. He received the Ph.D. degree in mechanical engineering from Xi'an Jiaotong University in 2002. He was a summer intern and then visiting student in Microsoft Research Asia in 1999 and 2000 respectively. He was invited to be a research assistant of Department of Computer Science, Hong Kong University of Science and Technology from August 2001 to February 2002. He joined GE Global Research as a Research & Development Engineer in January 2003. His research interests include signal processing, machine learning and process monitoring.

YAONAN WANG received the B.S. degree in Computer Engineering from East China Technology Institute, China in 1981. He received the M.S. and Ph.D. degrees in Electrical Engineering from Hunan University, China, in 1990 and 1994, respectively. From 1994 to 1995 he was a Postdoctoral Research Fellow in National University of Defense Technology. From 1981 to 1994 he worked in East China Technology Institute. Since 1995, he has been a professor in Hunan University, China. His research interests are image processing, pattern recognition,and industrial process control.

Figure Captions

Fig. 1. One stage of 2-D DWT and inverse DWT. Here, $2\downarrow 1$ denotes keeping one column out of two, and $1\downarrow 2$ denotes keeping one row out of two. Similarly, $2\uparrow 1$ denotes putting one column of zeros between each column, and $1\uparrow 2$ denotes putting one row of zeros between each row.

Fig. 2. One stage of 2-D DWFT and inverse DWFT. Here, $h\uparrow 2^i$ denotes inserting $2^i - 1$ zeros between every sample in filter h, and similarly for $g\uparrow 2^i, \tilde{h}\uparrow 2^i$ and $\tilde{g}\uparrow 2^i$.

Fig. 3. Image decompositions for DWT and DWFT.

Fig. 4. An example of separable data. Patterns from the two classes are represented by $\times$ and $\circ$ respectively. The extracted support vectors are shown in bold.

Fig. 5. The 30 natural textures used in the experiment. First row: D1, D3, D6, D11, D16 (left to right). Second row: D17, D20, D21, D24, D28. Third row: D29, D32, D34, D35, D46. Fourth row: D47, D49, D51, D52, D53. Fifth row: D55, D56, D57, D65, D78. Sixth row: D82, D84, D85, D101, D104.

Fig. 6. Average classification accuracy vs the number of training samples.

Fig. 7. Classification accuracies for each texture class. Textures are ordered by increasing performance of the fused SVM. The number of decomposition levels is 5, and 10% of the samples are used for training.

Tables

Table 1
Average classification accuracy vs the number of training samples, with 3 decomposition levels. Numbers in bold indicate the highest accuracy obtained over all the methods tested, and results marked with asterisks indicate that the fused SVM performs significantly better than the other methods at the 0.01 level of confidence according to the paired t-test [54].

proportion of samples used for training	Bayes distance	Mahalanobis distance	LVQ	SVM ($\sigma = 0.1$)	SVM ($\sigma = 1$)	SVM ($\sigma = 5$)	fused SVM
1.25%	69.91%	70.05%	65.90%	**76.01%**	71.96%	72.43%	74.59%
2.5%	77.86%	78.42%	76.65%	84.68%	84.91%	84.72%	**86.79%***
3.75%	79.74%	80.50%	81.67%	81.18%	83.35%	82.89%	**84.16%***
5%	83.20%	83.90%	82.94%	87.58%	90.90%	88.93%	**91.29%***
6.25%	84.89%	85.67%	86.28%	85.17%	87.40%	85.66%	**88.09%***
7.5%	86.55%	87.28%	87.28%	87.72%	90.65%	91.52%	**91.90%***
8.75%	86.87%	87.66%	88.02%	89.24%	91.81%	91.63%	**92.53%***
10%	87.83%	88.37%	87.81%	90.40%	93.68%	94.03%	**94.32%***

Table 2
Average classification accuracy vs the number of training samples, with 4 decomposition levels.

proportion of samples used for training	Bayes distance	Mahalanobis distance	LVQ	SVM ($\sigma = 0.1$)	SVM ($\sigma = 1$)	SVM ($\sigma = 5$)	fused SVM
1.25%	79.07%	**79.60%**	73.12%	75.38%	71.67%	76.55%	76.53%
2.5%	85.59%	85.76%	81.22%	83.76%	83.73%	85.11%	**86.23%***
3.75%	88.01%	87.32%	84.91%	86.59%	87.13%	88.78%	**89.14%***
5%	90.44%	89.83%	88.00%	89.34%	90.16%	90.91%	**91.46%***
6.25%	90.86%	90.51%	89.32%	90.82%	92.41%	92.58%	**93.23%***
7.5%	91.97%	91.32%	90.79%	91.35%	93.02%	93.23%	**93.78%***
8.75%	92.45%	92.14%	90.98%	91.89%	93.67%	93.65%	**94.28%***
10%	92.81%	92.57%	91.91%	93.19%	94.60%	94.42%	**95.16%***

Table 3
Average classification accuracy vs the number of training samples, with 5 decomposition levels.

proportion of samples used for training	Bayes distance	Mahalanobis distance	LVQ	SVM ($\sigma = 0.1$)	SVM ($\sigma = 1$)	SVM ($\sigma = 5$)	fused SVM
1.25%	**79.47%**	78.68%	69.72%	74.27%	76.61%	78.21%	78.45%
2.5%	86.47%	86.19%	80.06%	**88.25%**	85.56%	87.52%	87.94%
3.75%	87.89%	87.69%	83.93%	86.57%	88.56%	89.61%	**90.18%***
5%	90.33%	90.23%	87.05%	90.14%	92.03%	91.78%	**92.51%***
6.25%	91.17%	91.23%	88.19%	89.99%	92.00%	91.38%	**92.96%***
7.5%	91.79%	91.89%	89.61%	92.74%	94.58%	94.61%	**95.07%***
8.75%	92.11%	92.14%	90.21%	94.31%	95.31%	94.98%	**95.90%***
10%	92.67%	92.62%	90.43%	94.19%	95.83%	95.81%	**96.34%***

Table 4
Individual classification accuracy for each texture class. The number of decomposition levels is 5, and 10% of the samples are used for training.

texture	Bayes distance	Mahalanobis distance	LVQ	fused SVM
D1	98.19%	98.44%	92.56%	**98.89%**
D101	99.69%	99.75%	94.31%	**100.00%**
D104	**99.69%**	99.53%	85.25%	97.50%
D11	95.39%	94.25%	93.97%	**99.44%**
D16	**100.00%**	**100.00%**	93.75%	**100.00%**
D17	96.25%	94.28%	85.97%	**99.89%**
D20	**100.00%**	**100.00%**	93.67%	**100.00%**
D21	**100.00%**	**100.00%**	94.28%	99.72%
D24	96.75%	95.17%	84.47%	**98.33%**
D28	61.25%	64.39%	**91.78%**	89.17%
D29	87.58%	87.83%	**94.25%**	89.44%
D3	90.83%	91.11%	85.47%	**97.78%**
D32	98.36%	98.67%	93.83%	**98.89%**
D34	99.61%	99.67%	92.92%	**99.72%**
D35	93.58%	93.47%	85.14%	**94.72%**
D49	**100.00%**	**100.00%**	87.83%	**100.00%**
D47	85.44%	85.36%	92.86%	**94.44%**
D51	97.44%	97.25%	86.67%	**97.50%**
D53	**100.00%**	**100.00%**	92.14%	**100.00%**
D55	97.14%	96.86%	93.03%	**98.89%**
D56	73.25%	73.31%	**83.72%**	75.19%
D6	**100.00%**	**100.00%**	94.06%	**100.00%**
D65	71.81%	75.08%	**93.75%**	91.39%
D52	99.06%	99.00%	83.44%	**100.00%**
D84	87.97%	88.75%	91.50%	**97.50%**
D78	**100.00%**	**100.00%**	92.00%	99.72%
D82	90.19%	92.22%	82.17%	**90.28%**
D85	89.22%	85.67%	93.11%	**98.17%**
D46	86.28%	84.28%	93.78%	**95.00%**
D57	85.00%	84.33%	81.22%	**88.61%**
overall	92.67%	92.62%	90.43%	**96.34%**