Kernel Fisher Discriminant for Texture Classification

Jianguo Zhang and Kai-Kuang Ma [†], *Senior Member, IEEE*

School of Electrical and Electronic Engineering

Nanyang Technological University, Singapore

Nanyang Avenue, 639798

[†]Email: ekkma@ntu.edu.sg

Abstract

Kernel Fisher discriminant (KFD) is a state-of-the-art nonlinear machine learning method, and it has great potential to outperform linear Fisher discriminant. In this paper, a nonlinear discriminative texture feature extraction method based on KFD is proposed for texture classification. We mathematically show that finding the optimal discriminative texture features is equivalent to finding the optimal discriminative projection directions of the input data by KFD. Our proposed KFD-based method integrates texture feature extraction, nonlinear dimensionality reduction, and discrimination in a unified framework. Optimized and closed-form solutions are derived for both two-class and multi-class texture classification problems, individually. Extensive experimental results clearly show that the proposed method yields excellent performance in texture classification and outperforms other texture recognition methods based on discriminative classifiers.

I. INTRODUCTION

Texture analysis plays an instrumental role in computer vision and image processing. It has received considerable amount of attentions in the past few decades, and numerous methods have been proposed for texture feature extraction and classification. Extensive applications of texture analysis involve medical imaging, remote sensing, industrial inspection, document segmentation, biometrics, texture-based image retrieval, and so on. Existing texture analysis methods can be broadly classified into four categories [1]: *statistical methods*, *model-based methods*, *signal* *processing methods*, and *structural methods*. Among them, the methods from the first three are commonly encounterd and will be reviewed as follows. From the purpose of feature extraction, texture features can be broadly divided into two categories: 1) texture features for *representation*; 2) texture features for *discrimination*. In the past, most of the texture features are extracted based on *representation*. A comparative study conducted by Randen *et al.* [2] suggests that it is more preferable to extract texture features based on *discrimination* instead of representation for texture classification. Hence learning discriminative texture features for texture classification is desirable.

Statistical methods are motivated from Julesz's findings that human visual systems usually recognize textured objects based on the statistical distribution of their image gray levels via the first-order, the second-order, or the higher-order statistics [3][4][5]. The most commonly used and referenced method is the so-called *gray-level co-occurrence matrix* (GLCM) pioneered by Haralick [6], which estimates texture properties related to the second-order statistics. It is worth pointing out that although statistical texture features are usually used to classify texture regions, most of them are extracted explicitly or implicitly based on statistical texture representation.

For *model-based methods*, textures are often viewed as mathematical image perceptual models. The key problems of these methods are how to choose a suitable model for characterizing the selected textures and how to estimate the parameters of these models based on some criteria. Another concern is that intensive computations are usually required to determine the proper parameters. The derived model parameters are used as the features to capture the perceived essential qualities of texture. Commonly-used texture models are autoregressive model (AR) [2], Markov random field (MRF) [7], and Wold decomposition model [8]. AR model is a linear model established from the training samples through the least-mean-squares fitting. MRF attempts to describe the relationship of texture pixels by the definition of cliques within a region of interest. Its optimization is based on maximizing the posteriori probability. Wold decomposition model is a perceptual texture model that decomposes textures into deterministic and non-deterministic fields, which correspond to regular texture component and random texture component, respectively. Usually, these models can capture the local contextual information in a textured image. However, the model parameters are optimized based on image representation instead of image discrimination.

Signal processing methods provide another set of powerful approaches for texture analysis, such as multichannel Gabor filter, wavelet transform, finite impulse response (FIR) filter, etc. Gabor filter is appealing because of its simplicity and support from neurophysiological experiments [9].

Jain *et al.* use Gabor filters for texture segmentation despite that it is designed based on texture reconstruction [10]. In texture filtering methods, a general filter bank is often too large because it is designed to capture general texture properties. However, textures can be classified by only a small set of filters, which gives rise to the filter selection problem. For this problem, Jain *et al.* employ neural network to select a minimum set of Gabor filters for texture discrimination while keeping the performance at an acceptable level compared to the case without filters selection [11]. In these filtering methods, texture images are usually decomposed into several feature images through projection by using a set of selected filters. These filters are often designed based on representation such that textures could be reconstructed with minimum information loss. On the other hand, our proposed approach is to extract features that could maximize the separation or discrimination among different textures.

For the purpose of pattern discrimination, *linear Fisher discriminant* (LFD) is a well-known linear discrimination technique which can incorporate feature extraction, dimensionality reduction and discrimination together. However its linear optimal solution heavily depends on the assumption that the input patterns have equal covariance matrix. This assumption is usually not true for real-world data. To overcome this limitation, a kernel version of Fisher discriminant is recently developed for the *nonlinear* discriminative feature extraction. The optimal solution of *kernel Fisher discriminant* (KFD) corresponds to the optimal Bayesian classifier which accounts for the minimization of the classification (Bayesian) error rate [12] [13].

In this paper, a nonlinear texture feature extraction method based on KFD is proposed. Accordingly, a texture classification method is also described. KFD is originally proposed for two-class problem [13]. Hence, extending KFD for multi-class texture classification is desirable. In this paper, we generalize it from two classes to multiple classes to deal with multi-class texture classification problem (denoted as *multi-class KFD*). For two-class problem, KFD finds only one projection direction followed by one-dimensional texture features. For a c-class problem, KFD simultaneously finds out $c-1$ optimal projection directions resulting in $c-1$ dimensional texture features via one-shot optimization. Unlike other texture feature extraction methods, the proposed KFD-based texture classification method optimizes the texture discrimination mask and performs feature extraction, nonlinear dimensionality reduction, and classification simultaneously. Unlike AR texture features, the proposed method is optimized based on discriminant instead of texture representation. Compared to multichannel texture filters such as Gabor or wavelet filter, KFD performs nonlinear feature dimensionality reduction based on discrimination. Moreover, the 'kernel trick' introduced in the proposed method ensures that the optimization does not depend on those undesirable assumptions of the input data required by LFD. All of the above-mentioned merits ensure the proposed method yielding high performance for both two-class and multi-class texture classification problems.

II. Texture Vector Construction

It is well-known that texture is not only characterized by the gray level at a given pixel but also the gray-level pattern in a neighborhood region surrounding the pixel. Based on this intuition, given several classes of texture patterns, we denote texture pattern j by $x_j(m,n)$ where m and n are the spatial indexes of the texture pattern. Accordingly, what we want to compute is a *discrimination mask* $\omega(m,n)$ (or, *filter mask*, as commonly called in the literature), and convolve it with $x_j(m,n)$. That is,

$$\begin{aligned} y_j(m,n) &= \omega(m,n) * x_j(m,n) \\ &= \sum_{p=0}^{R-1}\sum_{q=0}^{C-1}\omega(p,q)\ x_j(m-p,n-q) \end{aligned} \tag{1}$$

where $*$ denotes the convolution operator. $\omega(m,n)$ is of size $R \times C$, and $y_j(m,n)$ is the output feature for texture pattern $x_j(m,n)$. The discrimination mask can be of any shape, and a rectangular region of support is the most commonly used one. For simplicity, we also adopt the rectangular mask in this paper, while other mask configurations can be easily extended. Using the lexicographically ordering of $\omega(m,n)$ and the local texture pattern of size $R \times C$ at position (m,n) by rows, the vector formulations can be obtained:

$$\boldsymbol{\omega} = \begin{bmatrix} \omega(0,0) \\ \vdots \\ \omega(0,C-1) \\ \omega(1,0) \\ \vdots \\ \omega(1,C-1) \\ \vdots \\ \omega(R-1,C-1) \end{bmatrix} ; \quad \mathbf{x}_j(m,n) = \begin{bmatrix} x_j(m,n) \\ \vdots \\ x_j(m,n-C+1) \\ x_j(m-1,n) \\ \vdots \\ x_j(m-1,n-C+1) \\ \vdots \\ x_j(m-R+1,n-C+1) \end{bmatrix} . \tag{2}$$

Thus, the feature extracted by the discrimination mask $\boldsymbol{\omega}$ at position (m,n) can be re-written as follows:

$$y_j(m,n) = \boldsymbol{\omega}^T \mathbf{x}_j(m,n). \tag{3}$$

Therefore, (3) tells us that a feature value for a local texture pattern at position (m, n) can be computed from the dot product between the mask coefficients and the corresponding pixel values. Most of the texture feature extraction methods such as AR model, Gabor filter, Laws filter [2], wavelet transform, with some modifications can be fit into the general vector formulation [14]. The key issue of texture discrimination is to find an optimal discrimination mask, which ensures that the output (texture features) of each texture class could result in the maximum separation between different textures such that promising texture classification results can be achieved. In the following sections, we will show how to derive an optimized discrimination mask by exploring the principle of KFD.

III. Kernel Fisher Discriminant

A. Linear Fisher Discriminant

Linear Fisher discriminant (LFD) is a well-known two-class discriminative technique. It aims to find the optimal projection direction such that the distance between the two mean values of the projected classes is maximized while each class variance is minimized. Thus LFD is capable of performing feature dimensionality reduction for classification, because only one-dimensional features are extracted for two-class problem. The following overviews the principle of LFD, which will be further extended into a kernel version (i.e., *nonlinear* Fisher discriminant). In the following, let us begin with two-class texture classification.

Let $\chi_1 = \mathbf{x}_1^1, ..., \mathbf{x}_{l_1}^1$ and $\chi_2 = \mathbf{x}_1^2, ..., \mathbf{x}_{l_2}^2$ be the sets of training samples generalized from two different texture classes. Each sample here is a vector which is formulated from texture pattern according to (2). l_1 and l_2 are the number of training samples corresponding to each class. Let l be the total number of training samples of all classes. LFD is given by the vector $\boldsymbol{\omega}$ (based on the formulation stated in Section II, $\boldsymbol{\omega}$ here can be alternatively considered as the discrimination mask) which maximizes the following Rayleigh coefficients [15]

$$J(\boldsymbol{\omega}) = \frac{\boldsymbol{\omega}^T \mathbf{S}_B \boldsymbol{\omega}}{\boldsymbol{\omega}^T \mathbf{S}_W \boldsymbol{\omega}} \tag{4}$$

where

$$\begin{aligned} \mathbf{S}_B &= (\mathbf{m}_1 - \mathbf{m}_2)(\mathbf{m}_1 - \mathbf{m}_2)^T \\ \mathbf{S}_W &= \sum_{j=1}^{2} \sum_{i=1}^{l_j} (\mathbf{x}_i - \mathbf{m}_j)(\mathbf{x}_i - \mathbf{m}_j)^T \end{aligned} \tag{5}$$

are between-class and within-class scatter matrices, respectively. $\mathbf{m}_j$ is defined by

$$\mathbf{m}_j = \frac{1}{l_j}\sum_{i=1}^{l_j}\mathbf{x}_i^j. \tag{6}$$

The optimal discrimination mask can be computed explicitly in a closed form by the following:

$$\begin{aligned}\boldsymbol{\omega}^* &= arg \ \max_{\boldsymbol{\omega}} J \\ &= \mathbf{S}_W^{-1}(\mathbf{m}_1 - \mathbf{m}_2)\end{aligned} \tag{7}$$

$\boldsymbol{\omega}^*$ is an optimal linear feature extractor. In this case, the optimal discriminative texture features can be directly computed using $\boldsymbol{\omega}^*$ by the projection defined in (3).

B. *Kernel Fisher Discriminant*

LFD has a very good connection to the optimal linear Bayesian classifier in the sense that the optimal projection direction corresponds to the optimal Bayesian classifier [12]. However, its optimality heavily depends on the assumption that all the classes have equal covariance matrices. It is obvious that the real-world data are usually not linearly separable and do not meet such a requirement. To overcome this limitation, *kernel Fisher discriminant* (KFD) has been introduced to find a nonlinear projection direction for two-class problem [13]. Its implementation can be achieved by employing the ’kernel trick’ introduced by Vapnik [16]. Accordingly, $\boldsymbol{\omega}$ becomes a nonlinear texture discrimination mask.

Consider there is a feature mapping ϕ which maps the input data into a higher-dimensional inner-product space F, that is, $\phi : \chi \rightarrow F$. Consequently, LFD can be applied in F (corresponding to *nonlinear* operation in the input space χ). It is equivalent to maximizing the following criterion:

$$J(\boldsymbol{\omega}) = \frac{\boldsymbol{\omega}^T S_B^{\phi}\boldsymbol{\omega}}{\boldsymbol{\omega}^T S_W^{\phi}\boldsymbol{\omega}} \tag{8}$$

where $\boldsymbol{\omega} \in F$. S_B^{ϕ} and S_W^{ϕ} are the corresponding between-class and within-class scatter matrices, respectively, formed in F, i.e.,

$$\begin{gathered} S_B^{\phi} = (\mathbf{m}_1^{\phi} - \mathbf{m}_2^{\phi})(\mathbf{m}_1^{\phi} - \mathbf{m}_2^{\phi})^T \\ S_W^{\phi} = \sum_{j=1}^{2}\sum_{i=1}^{l_j}(\boldsymbol{\phi}(\mathbf{x}_i) - \mathbf{m}_j^{\phi})(\boldsymbol{\phi}(\mathbf{x}_i) - \mathbf{m}_j^{\phi})^T \end{gathered} \tag{9}$$

with $\mathbf{m}_j^{\phi} = \frac{1}{l_j}\sum_{i=1}^{l_j}\boldsymbol{\phi}(\mathbf{x}_i^j)$.

From the theory of reproducing kernels [17], the solution of $\boldsymbol{\omega} \in F$ must lie in the span of all the training samples in F. Thus, $\boldsymbol{\omega}$ can be formed by a linear combination of the mapped training samples in F as follows:

$$\boldsymbol{\omega} = \sum_{i=1}^{l} \alpha_i \boldsymbol{\phi}(\mathbf{x}_i). \tag{10}$$

By combining (10) and the definition of $\mathbf{m}_j^{\phi}$, we can compute the projection between the two vectors in F as follows:

$$\boldsymbol{\omega}^T \mathbf{m}_j^{\phi} = \frac{1}{l_j} \sum_{i=1}^{l} \sum_{k=1}^{l_j} \alpha_i \langle \boldsymbol{\phi}(\mathbf{x}_i), \boldsymbol{\phi}(\mathbf{x}_k^j) \rangle \tag{11}$$

where $\langle \boldsymbol{\phi}(\mathbf{x}_i), \boldsymbol{\phi}(\mathbf{x}_k^j) \rangle$ represents the inner product between $\boldsymbol{\phi}(\mathbf{x}_i)$ and $\boldsymbol{\phi}(\mathbf{x}_k^j)$. By introducing a kernel function $k(\mathbf{x}, \mathbf{y})$ to represent the inner product $\langle \mathbf{x}, \mathbf{y} \rangle$ in F, (11) can be rewritten as follows:

$$\boldsymbol{\omega}^T \mathbf{m}_j^{\phi} = \frac{1}{l_j} \sum_{i=1}^{l} \sum_{k=1}^{l_j} \alpha_i k(\mathbf{x}_i, \mathbf{x}_k^j) = \boldsymbol{\alpha}^T \boldsymbol{\mu}_j \tag{12}$$

where $\boldsymbol{\mu}_j^i = \frac{1}{l_j} \sum_{k=1}^{l_j} k(\mathbf{x}_i, \mathbf{x}_k^j)$. Thus by using the definition of $\mathbf{S}_B^{\phi}$ in (9) and (12), the numerator of (8) can be computed as follows:

$$\boldsymbol{\omega}^T \mathbf{S}_B^{\phi} \boldsymbol{\omega} = \boldsymbol{\alpha}^T \mathbf{M} \boldsymbol{\alpha} \tag{13}$$

where $\mathbf{M} = (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)^T$. Similarly, the denominator of (8) can be reformulated as follows:

$$\boldsymbol{\omega}^T \mathbf{S}_W^{\phi} \boldsymbol{\omega} = \boldsymbol{\alpha}^T \mathbf{N} \boldsymbol{\alpha} \tag{14}$$

where $\mathbf{N} = \sum_{j=1,2} \mathbf{K}_j (\mathbf{I} - \mathbf{L}_j) \mathbf{K}_j^T$, and $\mathbf{K}_j$ is an $l \times l_j$ kernel matrix of class j with $(\mathbf{K}_j)_{nm} = k(\mathbf{x}_n, \mathbf{x}_m^j)$. $\mathbf{I}$ is the identity matrix and $\mathbf{L}_j$ is the matrix with all entries $l_j{}^{-1}$. Thus, the optimization of (8) is equivalent to finding the optimal value of $\boldsymbol{\alpha}$ by maximizing the following criterion:

$$J(\boldsymbol{\alpha}) = \frac{\boldsymbol{\alpha}^T \mathbf{M} \boldsymbol{\alpha}}{\boldsymbol{\alpha}^T \mathbf{N} \boldsymbol{\alpha}} \tag{15}$$

The optimal vector $\boldsymbol{\alpha}^*$ can be computed by finding the leading eigenvector of $\mathbf{N}^{-1}\mathbf{M}$. Once $\boldsymbol{\alpha}^*$ is obtained, the projection of a test pattern $\mathbf{x}_t$ onto $\boldsymbol{\omega}$ can be computed by

$$\langle \boldsymbol{\omega}, \boldsymbol{\phi}(\mathbf{x}_t) \rangle = \sum_{i=1}^{l} \alpha_i k(\mathbf{x}_i, \mathbf{x}_t) \tag{16}$$

Note that rather than computing the left-hand side of (16), the right-hand side can be much more easily obtained via a linear combination of the inner products which is independent of the mapping operator ϕ. This tells us that we only need to define a kernel form of an inner product instead of computing the explicit form of this mapping. Without considering the mapping ϕ explicitly, KFD can be constructed by selecting the proper kernel. Commonly-used kernel functions are summarized as follows:

- Gaussian kernel (*radial basis function* (RBF)): $k(\mathbf{x}, \mathbf{y}) = \exp\left(\frac{-\|\mathbf{x}-\mathbf{y}\|^2}{\sigma}\right)$
- Polynomial kernel: $k(\mathbf{x}, \mathbf{y}) = (\mathbf{x} \cdot \mathbf{y})^n$
- Tangent hyperbolic kernel: $k(\mathbf{x}, \mathbf{y}) = \tanh(\mathbf{x} \cdot \mathbf{y} + \Theta)$

where σ, n, and Θ are the parameters of the three kernels, respectively. As the dimensionality of F is usually much higher than the number of the training samples which could cause the matrix $\mathbf{N}$ being non-positive definite, consequently, finding the optimal value of $\boldsymbol{\alpha}$ is an ill-posed problem. The commonly-used approach to solve this problem is to employ *regularization*, which simply adds a multiple of the identity matrix or the kernel matrix $\mathbf{K}$ to $\mathbf{N}$ to guarantee that $\mathbf{N}$ is positive definite. In this paper, we use the former as in [13]. That is,

$$\mathbf{N}_r = \mathbf{N} + r\mathbf{I} \tag{17}$$

IV. TWO-CLASS TEXTURE CLASSIFICATION

KFD is originally designed for the problem of binary classification. For the two-class texture classification, the optimal discrimination mask is derived by finding the optimal value $\boldsymbol{\alpha}^*$. Fortunately, the closed-form solution of maximizing (15) can be computed as follows:

$$\boldsymbol{\alpha}^* = arg \max_{\boldsymbol{\alpha}} J(\boldsymbol{\alpha}) = \mathbf{N}^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2) \tag{18}$$

$$\boldsymbol{\omega}^* = \sum_{i=1}^{l} \phi(\mathbf{x}_i). \tag{19}$$

The optimal discrimination mask can not be computed explicitly, because we do not know the explicit form of mapping ϕ. Since our objective is to derive the features by projecting the texture patterns onto the discrimination mask defined by (3), we do not have to compute the explicit form of $\boldsymbol{\omega}$ but the projection directly. Fortunately, given a texture pattern, its one-dimensional feature can be directly extracted by the following equation without using the explicit form of the mapping function ϕ:

$$y = \mathbf{N}^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2) \cdot \mathbf{K}_t \tag{20}$$

where $\mathbf{K}_t^i = k(\mathbf{x}_i, \mathbf{x}_t)$. Since the KFD is optimized via the maximum separation, which accounts for the minimum classification (Bayesian) error according to the probability theory, it has fully utilized the discriminative information expressed by the training samples. Therefore, the selection of the classifiers for classifying the obtained texture features is not an issue; thus any classifier can be the candidate. For simplicity, in this paper, we use the simplest nearest-mean classifier for classification [18]. Using y_j $(j = 1, 2)$ to represent the outputs of KFD for the training samples of texture class j, the mean of each class $\overline{y}_j$ can be computed. Given a test texture pattern t, its corresponding KFD feature is y_t. Thus the decision function is simply defined as follows:

$$f(t) = arg \min_j |y_t - \overline{y}_j|. \tag{21}$$

V. Multi-class texture classification

In the previous sections, we have discussed the problem of two-class texture classification based on KFD. However, many real applications involve more than two kinds of texture. Therefore, in this section, we generalize the two-class KFD algorithm to deal with the multi-class texture classification. More than one filters are usually needed in the multi-class texture classification. Suppose that we have c texture classes and each class has l_j training samples. In this case, $\boldsymbol{\omega}$ becomes a projection matrix which contains $c-1$ directions. To be more clear, hence, we define $\mathbf{W}$ to represent such a matrix, and $\boldsymbol{\omega}$ is a column of $\mathbf{W}$. Multi-class KFD is a natural generalization of the kernel Fisher discriminant based on multiple discriminants [15]. In F, we rewrite the Fisher criterion as follows:

$$J(\mathbf{W}) = \frac{|\ \mathbf{W}^T S_B^{\phi} \mathbf{W}\ |}{|\ \mathbf{W}^T S_W^{\phi} \mathbf{W}\ |} \tag{22}$$

$$\begin{aligned} S_B^{\phi} &= \sum_{j=1}^{c} l_j (\mathbf{m}_j^{\phi} - \mathbf{m}^{\phi})(\mathbf{m}_j^{\phi} - \mathbf{m}^{\phi})^T \\ S_W^{\phi} &= \sum_{j=1}^{c} \sum_{i=1}^{l_j} (\boldsymbol{\phi}(\mathbf{x}_i) - \mathbf{m}_j^{\phi})(\boldsymbol{\phi}(\mathbf{x}_i) - \mathbf{m}_j^{\phi})^T \end{aligned} \tag{23}$$

with $\mathbf{m}^{\phi} = \frac{1}{l} \sum_{i=1}^{l} \boldsymbol{\phi}(\mathbf{x}_i)$, and $\mathbf{m}_j^{\phi} = \frac{1}{l_j} \sum_{i=1}^{l_j} \boldsymbol{\phi}(\mathbf{x}_i)$, where $j = 1, ..., c$. Referring to (22), any column of the solution $\mathbf{W}$ must lie in the span of all training samples. Using $\boldsymbol{\omega}$ to represent any one column of $\mathbf{W}$, we have:

$$\boldsymbol{\omega} = \sum_{i=1}^{l} \alpha_i \boldsymbol{\phi}(\mathbf{x}_i). \tag{24}$$

Similarly as the two-class kernel Fisher discriminant, we can project each of the class mean onto an axis of F by using only the dot product:

$$\boldsymbol{\omega}^T \mathbf{m}_j^{\phi} = \frac{1}{l_j} \sum_{i=1}^{l} \sum_{k=1}^{l_j} \alpha_i k(\mathbf{x}_i, \mathbf{x}_k^j) = \boldsymbol{\alpha}^T \boldsymbol{\mu}_j. \tag{25}$$

It follows that

$$\begin{aligned} \boldsymbol{\omega}^T S_B^{\phi} \boldsymbol{\omega} &= \boldsymbol{\alpha}^T \mathbf{M} \boldsymbol{\alpha} \\ \boldsymbol{\omega}^T S_W^{\phi} \boldsymbol{\omega} &= \boldsymbol{\alpha}^T \mathbf{N} \boldsymbol{\alpha} \end{aligned} \tag{26}$$

where $\mathbf{M} = \sum_{j=1}^{c} l_j (\boldsymbol{\mu}_j - \boldsymbol{\mu})(\boldsymbol{\mu}_j - \boldsymbol{\mu})^T$, $\mathbf{N} = \mathbf{K}\mathbf{K}^T - \sum_{j=1}^{c} l_j \boldsymbol{\mu}_j \boldsymbol{\mu}_j^T$, $\boldsymbol{\mu}^i = \frac{1}{l} \sum_{k=1}^{l} k(\mathbf{x}_i, \mathbf{x}_k)$, and $\boldsymbol{\mu}_j^i = \frac{1}{l_j} \sum_{k=1}^{l_j} k(\mathbf{x}_i, \mathbf{x}_k^j)$. Thus, the goal of multi-class KFD is to find

$$\mathbf{A}^* = arg \max_{\mathbf{A}} \frac{| \mathbf{A}^T \mathbf{M} \mathbf{A} |}{| \mathbf{A}^T \mathbf{N} \mathbf{A} |} \tag{27}$$

where $\mathbf{A} = [\boldsymbol{\alpha}_{\mathbf{1}}, ..., \boldsymbol{\alpha}_{\mathbf{c-1}}]$. The computation of $\mathbf{M}$ and $\mathbf{N}$ requires only kernel computations. The optimal matrix $\mathbf{A}^*$ can be computed by finding the $(c-1)$ leading eigenvectors of $\mathbf{N}^{-1}\mathbf{M}$ corresponding to the none-zero eigenvalues, which are equivalent to the optimization of the well-known Rayleigh coefficients. Once $\mathbf{A}^*$ is obtained, for a given texture pattern $\mathbf{x}_t$, we can map it to a $(c-1)$ dimensional discrimination space spanned by the columns of $\mathbf{A}^*$. This projection can be computed as follows:

$$\mathbf{y} = \mathbf{A}^{*T} \mathbf{K}_t \tag{28}$$

where $K_t^i = k(\mathbf{x}_i, \mathbf{x}_t)$, and $\mathbf{y}$ is a feature vector of size $(c-1) \times 1$. Similarly, as the two-class texture classification problem, the decision function for a given texture pattern t thus becomes the following form:

$$f(t) = arg \min_{t} D(\mathbf{y}_t - \overline{\mathbf{y}}_j) \tag{29}$$

where $\overline{\mathbf{y}}_j$ is the mean feature vector of class j. $D(\cdot)$ is the distance function of the feature vector of $\mathbf{y}_t$ and $\overline{\mathbf{y}}_j$. In this paper, we simplify it as the Euclidean distance.

Usually, more different textures involve higher computational complexity. For example, by using typical one-against-the-rest technique with two-class KFD, for a c-class texture classification problem, the computation cost will be c times of that of the two-class texture classification problem, since we have to find c filters by two-class optimization, each of which discriminating one texture against the rest. Thus we need c times two-class optimizations when using typical one-against-the-rest technique. However, within the framework of the multi-class KFD, the optimization is only one shot and furthermore the solution is in a closed form. The complexity of the computation of multi-class texture classification is thus reduced almost as the same as that of two-class texture classification.

VI. Post-Processing

Classification map often contains speckle noise, resulted from mis-classifications. Applying a post-processing technique (e.g., morphological filtering [11]) over the map is able to effectively remove them in order to get a higher classification accuracy. One concern of using this technique is how to choose a proper window size for it will introduce a tradeoff between minimizing the total error and identifying the border between two textures correctly. The improvement of morphological filtering on the final results depends on the size and the shape of uniform texture regions. We experimentally set the size of the filtering window as 5×5. An average improvement of 5% is achieved in terms of the error rate with respect to the ground truth segment. In this paper, we focus on the measurement of classification error as a yardstick of performance evaluation. The reported error rates are computed based on the decision function in (29) without applying any post-processing.

VII. Experimental Results

In our experiments, the efficacy and effectiveness of KFD-based texture feature extraction approach for texture segmentation are evaluated by performing supervised segmentation based on several test images with various texture complexities. The test images are constructed based on the textures from two commonly-used natural texture image databases: Brodatz album [19] and MIT Vis-Tex database [20]. Each texture image has a size of 512×512, with 8 bits per pixel (i.e., 256 grey levels). Each image is globally histogram equalized and normalized to [-1, +1] to ensure that the textures are not trivially discriminable simply based on the local mean or local variance. Different portions of the input patterns of each texture class are randomly selected and used for training KFD. We avoid using the texture patterns on the texture borders for training because these patterns are not suitable for training. The size of the texture-pattern window ($R \times C$ in (1)) will affect the classification performance. Therefore, the performance of our method for texture classification are tested on different textures with different window sizes, from 5×5 to 21×21 (all in odd numbers). The texture sources used in our experiments range from two-class texture images to as many as nine-class texture images as shown in Fig. 1 and Fig. 2, respectively.

A. Two-class Texture Image

Several texture pairs as shown in Fig. 1 are tested for binary classification using the proposed method. The size of these test images is 256×512. The training data are obtained by randomly

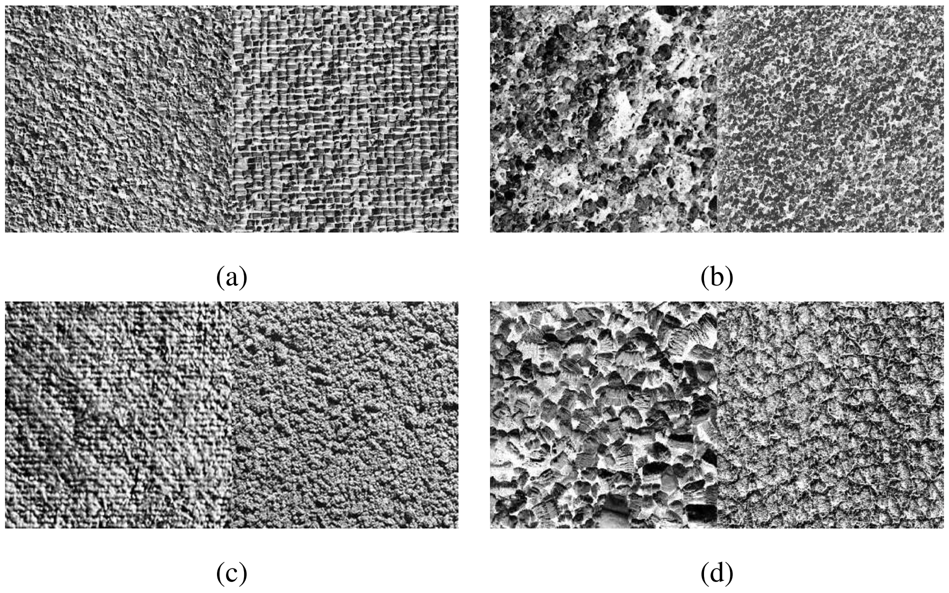


Fig. 1. Two-class texture images used in our experiments. (a) D4 and D84 from [19]; (b) D28 and D29 from [19]; (c) Fabric.0007 and Food.0005 from [20]; (d) D5 and D92 from [19].

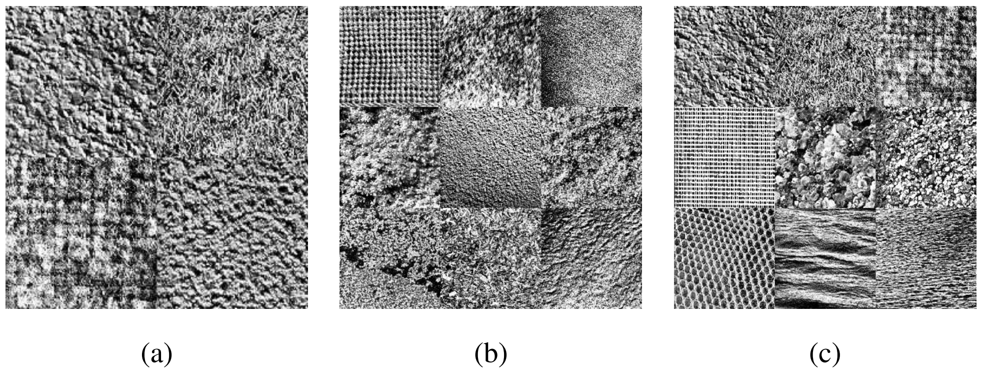


Fig. 2. Multi-class texture images used in our experiments. (a) D4, D9, D19, and D57 are from [19]; (b) Fabric.0009, Fabric.0016, Fabric.0019, Flowers.0005, Food.0005, Leaves.0003, Misc.0000, Misc.0002, and Sand.0000 from [20]; (c) D4, D9, D19, D21, D24, D28, D36, D37, and D38 from [19].

selecting 400 patterns from each texture class, which account for only 0.6% of the total available thousands of input texture patterns. That is, the KFD texture feature extraction is performed based on 800 texture patterns for each texture pair.

The implementation of KFD needs to select a proper kernel function and tune the corresponding kernel parameters. Two important kernels are Gaussian kernel and polynomial kernel. For polynomial kernel, the choice of its degree n will dramatically affect the classification performance. A set of experiments are conducted on the texture images in Fig. 1 with the window size of

17×17 and 19×19. Our experiments show that these window sizes are effective in capturing texture properties. Table I tabulates the error rates produced by using polynomial kernels with different degrees, and clearly shows that degree $n = 2$ achieves the best error rate for all test images as shown in Fig. 1. Thus, $n = 2$ is adopted in the follow-up experiments. It is worth to highlight that, for higher degrees, the classification performances are degraded. In these cases, we also note that the training errors is zero[1]. This can be interpreted by the over-fitting problem [12]. For Gaussian kernel, the parameter is selected as $\sigma = 0.3 \cdot N$ (where N is the dimensionality of the input patterns), which is also an optimal parameter setting for RBF-based support vector machine (SVM) [12][17].

For an overall evaluation of the effect of both polynomial and Gaussian kernels in the proposed method, another group of experiments are conducted. The effect of applying different sizes of window is also evaluated in this experiment. Table II summarizes the classification error rates by these two kernels. From the table, we can see that, for texture pairs shown in Fig. 1, the best error rates are achieved by using the windowing sizes of 17×17 and 19×19. This suggests that these window sizes are suitable to capture the texture properties.

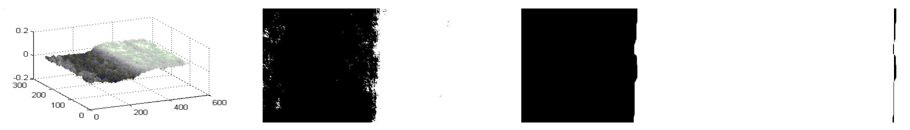
We observe that, for non-homogeneous textures, the classification accuracy of polynomial kernels is worse than that of RBF kernels. This can be demonstrated by the error rates shown in Table II. It can be seen that, for Figs. 1(b) and 1(d) with 17×17 window, the error rates with RBF kernel (5.7% and 8.6%, respectively) are significant lower than that with polynomial kernel (26.3% and 22.5%, respectively). The same can also be observed from the results using 19×19 window. The observation coincides with some research in kernel theory. It has been shown that, by using polynomial kernel, the distance of mapped training samples in F could become extremely large [12]. This property of polynomial kernel may distort the optimal direction found by KFD, especially for more scattered training samples in the input space resulted from non-homogeneous textures as shown in Figs. 1(b) and 1(d). This may lead to much degraded performance of polynomial kernel-based KFD on non-homogeneous textures.

On the contrary, from Table II, the results by Gaussian kernel are much more stable for both homogeneous and non-homogeneous textures in terms of error rates (3.7%, 5.2%, 7.1%, and 8.9% using 19×19 window). Moreover, more accurate results are achieved by Gaussian kernel than by polynomial kernel overall. To show the discriminative properties of KFD-based texture features for discrimination by using Gaussian kernel, we also plot the feature map of each texture

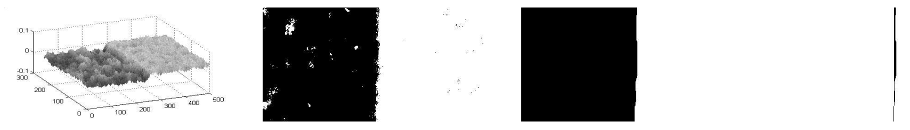
[1]Note that zero training errors does not mean accurate classification. This is overlooked in the literature [21].

pair, respectively, as shown in Fig. 3 (a.1), (b.1), (c.1), and (d.1). From these feature maps, we can see distinct discontinuity occurs at the boundary between two textures within each texture pair. After applying post-processing, the proposed method achieves fairly promising segmentation results. Note that, for texture pairs: D5 and D92, it is very hard to discriminate even by Gabor features [2][21]. However, our proposed method has achieved a rather low error rate of 8.6% using 17×17 window.

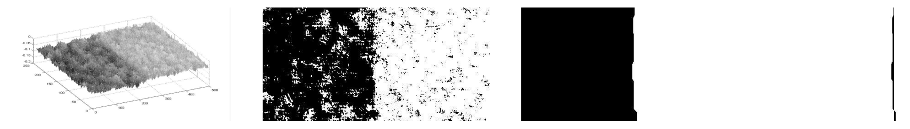
In conclusion, the Gaussian kernel outperforms polynomial kernel and yields a successful discrimination between closely-resemble textures. Therefore, it is a preferable choice of kernels for KFD-based texture classification.



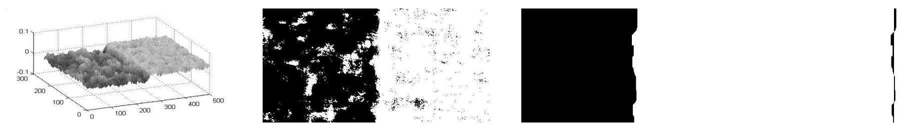
(a.1) (a.2) (a.3) (a.4)



(b.1) (b.2) (b.3) (b.4)



(c.1) (c.2) (c.3) (c.4)



(d.1) (d.2) (d.3) (d.4)

Fig. 3. Two-class segmentation results of texture images in Fig.3, where each row corresponds to Fig. 1(a), (b), (c), and (d), respectively. The first column corresponds to the feature maps of different texture pairs, respectively. The second column is the classification results. The third column is the classification results after post-processing. The fourth column is the error maps in terms of misclassified pixels.

B. Multi-class Texture Image

In this section, the performance of multi-class KFD in the context of multi-class texture images is evaluated. Three multi-class texture images, as shown in Fig. 2, are created for this experiment.

TABLE I

ERROR RATES (%) WITH 17×17 AND 19×19 WINDOWS USING DIFFERENT POLYNOMIAL DEGREES

	Fig. 1 (a)		Fig. 1 (b)		Fig. 1 (c)		Fig. 1 (d)	
Degree, n	17×17	19×19	17×17	19×19	17×17	19×19	17×17	19×19
1	49.6	49.7	46.8	50.0	47.1	47.6	48.1	47.7
2	**3.5**	**2.7**	**26.3**	**28.3**	**11.7**	**7.3**	**22.5**	**33.6**
3	19.7	20.7	49.6	49.6	49.5	49.0	49.7	49.6
4	12.9	16.1	49.6	49.6	43.7	38.0	50.0	50.0
5	34.1	34.9	49.6	49.6	49.6	49.6	49.6	49.6
6	30.1	30.2	49.6	49.6	49.6	49.6	49.8	49.6

TABLE II

ERROR RATES (%) OF TWO-CLASS TEXTURE IMAGES USING VARIOUS WINDOW SIZES

		Window Size								
Kernel	Image	5×5	7×7	9×9	11×11	13×13	15×15	17×17	19×19	21×21
RBF	Fig. 1 (a)	20.4	16.1	11.0	8.3	6.5	5.4	4.2	**3.7**	4.9
	Fig. 1 (b)	18.2	13.8	10.6	8.8	7.6	6.8	5.7	**5.2**	5.4
	Fig. 1 (c)	20.2	17.5	14.6	14.2	12.1	10.6	9.6	**7.1**	7.2
	Fig. 1 (d)	19.4	15.6	13.6	12.1	11.3	10.3	**8.6**	8.9	10.1
Poly ($n = 2$)	Fig. 1 (a)	25.5	33.4	19.5	11.6	8.0	5.3	3.5	**2.7**	2.6
	Fig. 1 (b)	22.8	30.7	24.2	23.0	23.1	24.2	**26.3**	28.3	29.7
	Fig. 1 (c)	15.8	22.5	18.3	18.7	17.6	15.9	11.7	**7.3**	7.8
	Fig. 1 (d)	23.0	22.6	24.5	26.4	24	26.3	**22.5**	33.6	34.8

Fig. 2(a) shows a 512×512 test image composed of four Brodatz textures. Figs. 2(b) and 2(c) are two 384×384 test images composed of nine texture images from Brodatz album and MIT Vis database, respectively. For all of these test images, 400 texture patterns are randomly selected from each texture class for training. Thus, a database of 1600 training samples are set up for Fig. 2(a), while a database of 3600 training samples for Figs. 2(b) and 2(c), respectively. Table III tabulates the error rates for each of the test images.

For Fig. 2, the lowest error rate obtained is 6.7% when using 19×19 window. Only three projection directions are needed to recognize all of these four textures in our proposed method. To understand the discrimination behavior of each feature component extracted by multi-class nonlinear texture filters (each column of $\mathbf{A}^*$) individually, the three feature maps produced by each filter are shown in Figs. 4(a)-(c), respectively. A very interesting observation is that each filter tries to discriminate one texture class from the others, while the combination of the three filters can give a successful discrimination among these four textures. A 3-D scatter plot is shown in Fig. 5, which presents some insights of the KFD texture features. Although these four textures are visually similar as shown in Fig. 4, a very distinct discrimination can be observed in the scatter plot. Hence, the four textures can be well separated by the extracted KFD features.

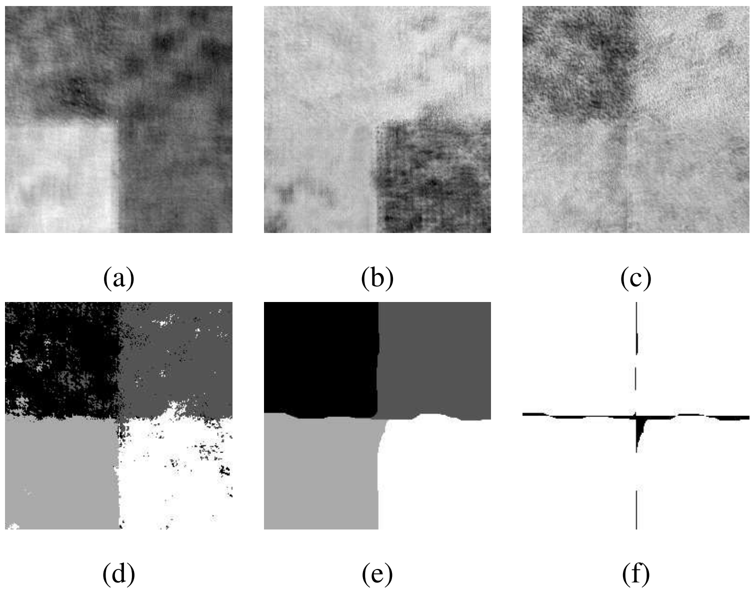


Fig. 4. Four-texture segmentation results for the test image Fig. 2 (a). (a)-(c) feature maps resulted from three feature components, individually; (d) classification results; (e) classification results after applying post-processing; (f) error map shown in terms of misclassified pixels.

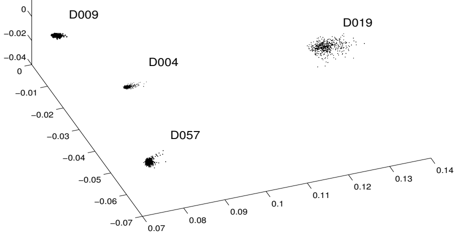


Fig. 5. 3-D scatter plot of KFD features for four-texture image in Fig. 2 (a).

TABLE III

ERROR RATES (%) OF MULTI-CLASS TEXTURE IMAGE AS SHOWN IN FIG. 2 UNDER DIFFERENT WINDOW SIZES USING RBF KERNEL

Image	Window Size								
	5×5	7×7	9×9	11×11	13×13	15×15	17×17	19×19	21×21
Fig. 2 (a)	23.5	15.2	11.2	8.3	7.4	6.8	6.8	**6.7**	7.3
Fig. 2 (b)	20.1	15.9	12.6	**11.4**	12.2	11.5	12.2	13.1	13.4
Fig. 2 (c)	30.1	23.9	20.7	20.8	17.8	16.6	14.1	13.9	**13.6**

Figs. 2(b) and 2(c) involve as many as nine textures. The increasing number of texture classes will increase the difficulty to discriminate these textures. Yet, in this case, satisfied error rates by using our proposed approach using RBF kernel are achieved as 11.4% for Fig. 2(b) using 11×11 window, 13.6% for Fig. 2(c) using 21×21 window. So the proposed method also performs well in multi-class texture classification.

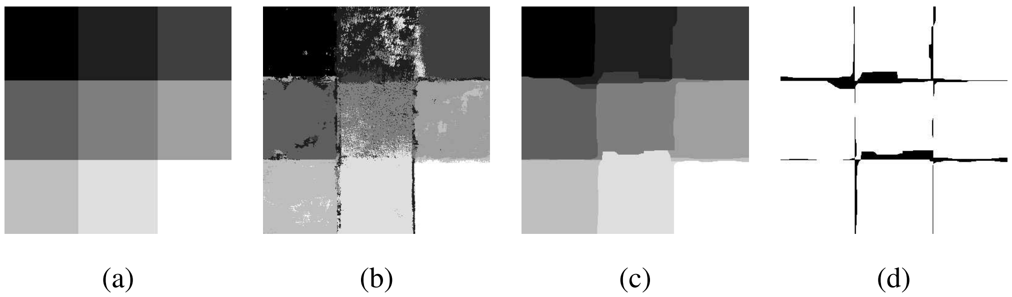


Fig. 6. Nine-Vis-texture segmentation results for the test image in Fig. 2(b): (a) ideal segmentation result; (b) classification result; (c) classification results after applying post-processing; (d) error map in terms of the misclassified pixels.

VIII. COMPARISON

In this section, a comparative study is further conducted with the SVM-based texture classification method [21]. Two types of commonly-used SVMs are investigated in the comparison: polynomial-based SVM and RBF-based SVM (denoted as poly-SVM and RBF-SVM, respectively). Texture classification with the two types of SVMs is performed with different settings of parameters. For RBF-SVM, the parameter of RBF width σ is set as $\sigma = \frac{1}{0.3*N}$ [12], where N is

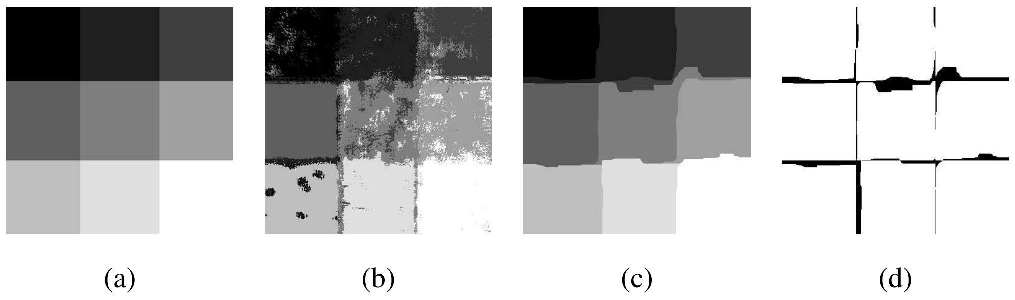


Fig. 7. Nine-Brad-texture segmentation results for the test image in Fig. 2 (c). (a) ideal segmentation result; (b) classification result; (c) classification result after applying post-processing; (d) error map in terms of the misclassified pixels.

the dimensionality of the input texture patterns. For poly-SVM, the degree of SVM is set as 5 for texture classification which is suggested by Kim *et al.* [21]. Accordingly, the regularization term C is set as 100 for both poly-SVM and RBF-SVM, respectively. The texture vector configurations follow the method presented in Section II. The implementation of SVM (LIBSVM) is used in this comparative study [22]. The 17×17 and 19×19 windows are selected for this study, because most of the lowest error rates are achieved with these windows, and they are suitable to capture texture properties as indicated in our previous study. Classification error rates with these methods are summarized in Table IV. Form this table, we can see that the performance of RBF-SVM is much better than that of poly-SVM (degree equals 5) in terms of classification error rates. RBF-KFD gives the lowest error rates among all of these methods listed in Table IV. In order to understand the merits of our proposed kernel-based texture discrimination method, we also incorporate the classification results by using the LFD-based texture classification method. We can see that the improvement by using the ”kernel trick” is very distinct. This also indicates that the textures studied in this paper are not linearly discriminable. Overall, the proposed KFD-based method outperforms the other texture discrimination methods in our comparison.

IX. CONCLUSION

In this paper, a nonlinear discriminative texture feature extraction method based on kernel Fisher discriminant (KFD) is proposed. Accordingly, a texture classification method based on the extracted texture features is described. The derived optimized closed-form solution gives a simple, powerful, and elegant way for both two-class and multi-class texture classification problems. Furthermore, compared to SVM-based texture classification method with different

TABLE IV

COMPARISON OF ERROR RATES (%) USING KFD-, SVM-, LFD-BASED TEXTURE CLASSIFICATION METHODS

Method	Fig. 1(a)		Fig. 1(b)		Fig. 1(c)		Fig. 1(d)	
	17×17	19×19	17×17	19×19	17×17	19×19	17×17	19×19
RBF-KFD	**4.2**	**3.7**	**5.7**	**5.2**	**9.6**	**7.1**	**8.6**	**8.9**
RBF-SVM	4.9	5.2	6.5	6.2	9.7	9.8	10.7	9.5
Poly-SVM ($p = 5$)	21.6	22.6	49.7	49.6	50.5	50.2	49.7	49.7
LFD	49.6	49.7	46.8	50.0	47.1	47.6	48.1	47.7

kernel functions, promising classification results with the smallest error rates are obtained by using the proposed method.

We also demonstrate that it is possible to extract good discriminative texture features without apply filtering on the image as a preprocessing step. Similar observations and arguments can be found in recent impressive works in [21] and [23].

At moment, our approach use rectangular region of support for each texture pixel. Rotation invariance can be achieved using circular region of support. It is also interesting to note that there are some recent work aiming to achieve invariance using local affine regions [24] [25]. The combination of such work and our approach is expected to achieve good performance in recognizing textures across a wide of separate views. Yet, this is not our focus of this paper. We will investigate this in our future research.

Note that the proposed method is a *supervised* approach. The automatic determination of the number of classes should also be considered in unsupervised classification. Hence, further study will be extended to *unsupervised* texture classification. Possible choices may include the use of k-means to automatically select the training samples for each texture class. Some of the applications of the proposed method may include object classification based on textures, document segmentation, and segmentation of remote sensing images, etc.

REFERENCES

[1] M. Tuceryan and A. Jain, “Texture analysis,” in *Handbook Pattern Recognition and Computer Vision*, C. Chen, L. Pau, and P. Wang (Eds.), Singapore: World Scientific, 1993, pp. 235–276.

[2] T. Randen and J. Husoy, “Filtering for texture classification: A comparative study,” *IEEE Trans. Pattern Analysis Machine Intelligence*, vol. 21, no. 4, pp. 291–310, 1999.

[3] B. Julesz, E. N. Gilbert, L. A. Shepp, and H. L. Frisch, “Inability of humans to discriminate between visual textures that agree in second-order statistics – revisited,” *Perception*, vol. I, no. 2, pp. 391–405, 1973.

[4] B. Julesz, "Visual pattern discrimination," *IRE Trans. on Information Theory*, vol. IT-8, no. 84-92, 1962.

[5] ——, "Experiments in the visual perception of texture," *Scientific American*, vol. 232, pp. 34–43, 1975.

[6] R. Haralick, K. Shangmugam, and L. Dinstein, "Textural features for image classification," *IEEE Trans. Systems, Man, and Cybernetics*, vol. 3, pp. 610–621, 1973.

[7] F. Cohen, Z. Fan, and M. Patel, "Classifcation of rotated and scaled textured images using gaussian markov random field models," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 13, no. 2, pp. 192–202, Feb. 1991.

[8] F. Liu and R. Picard, "Periodicity, directionality, and randomness: Wold features for image modeling and retrieval," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 18, no. 7, pp. 722 –733, July 1996.

[9] O. Faugeras, "Texture analysis and classification using a human visual model," *Proc. IEEE Int. Conf. Pattern Recognition, Kyoto*, pp. 549–552, 1978.

[10] A. Jain and F. Farrokhnia, "Unsupervised texture segmentation using Gabor filters," *Pattern Recognition*, vol. 24, no. 12, pp. 1167–1186, 1991.

[11] A. Jain and K. Karu, "Learning texture discrimination masks," *IEEE Trans. Pattern Analysis Machine Intelligence*, vol. 18, no. 2, pp. 195–205, Feb. 1996.

[12] B. Schölkopf and A. J. Smola, *Learning with Kernels: Support Vector Machines, Regularization, Optimization and Beyond*. Cambridge, MA: MIT Press, 2002.

[13] S. Mika, G. Rätsch, J. Weston, B. Schölkopf, and K.-R. Müller, "Fisher discriminant analysis with kernels," in *Neural Networks for Signal Processing*, Y.-H. Hu, J. Larsen, E. Wilson, and S. Douglas (Eds.), IEEE, 1999, pp. 41–48.

[14] T. Randen and J. Husy, "Texture segmentation using filters with optimized energy separation," *IEEE Trans. on Image Processing*, vol. 8, no. 4, pp. 571–582, April 1999.

[15] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification*. New York: Wiley-Interscience, 2001.

[16] V. Vapnik, *Statistical Learning Theory*. New York: John Wiley and Sons Inc., 1998.

[17] S. Mika, G. Rätsch, J. Weston, B. Schölkopf, and K.-R. Müller, "Constructing descriptive and discriminative nonlinear features: Rayleigh coefficients in kernel feature spaces," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 25, no. 5, pp. 623–628, May 2003.

[18] K. Fukunaga, *Introduction to statistical pattern recognition*. Academic press, Inc., Boston, 1990.

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

[20] MIT Vison and Modeling Group, http://www.media.mit.edu/vismod/, 1998.

[21] K. I. Kim, K. Jung, S. H. Park, and H. J. Kim, "Support vector machines for texture classification," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 24, no. 11, pp. 1542–1550, Nov. 2002.

[22] C.-C. Chang and C.-J. Lin, "Training u-support vector classifiers: Theory and algorithms," *Neural Computation*, no. 9, pp. 2119–2147, Sept. 2001.

[23] M. Varma and A. Zisserman, "Texture classification: Are filter banks necessary?" in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, vol. 2, June 2003, pp. 691–698.

[24] S. Lazebnik, C. Schmid, and J. Ponce, "A sparse texture representation using affine-invariant regions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, vol. II. Madison, WI, June 2003, pp. 319–324.

[25] ——, "A sparse texture representation using local affine regions," *Submitted to IEEE Trans. Pattern Analysis and Machine Intelligence*,

2004.