Effective Texture Classification by Texton Encoding Induced Statistical Features

Jin Xie[a], Lei Zhang[a,*], Jane You[a], Simon Shiu[a]

[a]*Department of Computing, The Hong Kong Polytechnic University, Hung Hom, Kowloon, Hong Kong, China*

Abstract

Effective and efficient texture feature extraction and classification is an important problem in image understanding and recognition. Recently, texton learning based texture classification approaches have been widely studied, where the textons are usually learned via K-means clustering or sparse coding methods. However, the K-means clustering is too coarse to characterize the complex feature space of textures, while sparse texton learning/encoding is time-consuming due to the l_0-norm or l_1-norm minimization. Moreover, these methods generally compute the texton histogram as the statistical features for classification, which may not be accurate enough. This paper presents an effective and efficient texton learning and encoding scheme for texture classification. First, a regularized least square based texton learning method is developed to learn the dictionary of textons class by class. Second, a fast two-step l_2-norm texton encoding method is proposed to code the input texture feature over the concatenated dictionary of all classes. Third, two types of histogram features are defined and computed from the texton encoding outputs: coding coefficients and coding residuals. Finally, the two histogram features are combined for classification via a nearest subspace classifier. Experimental results on the CUReT, KTH_TIPS and UIUC datasets demonstrated that the proposed method is very promising, especially when the number of available training samples is limited.

Keywords: Texture classification, texton learning, texture feature extraction, sparse representation

1. Introduction

Texture feature extraction and classification is an important research topic in many image processing, computer vision and pattern recognition tasks, including medical imaging, remote sensing, material identification and image classification, etc. In the past decades, a variety of texture classification methods [1, 7, 2, 3, 4, 5, 9, 10, 11, 12, 14, 28, 15, 16, 17] have been proposed. The use of co-occurrence matrices [1], which exploits the non-parametric statistics at the pixel level, is still popular in texture classification. As a simple and efficient statistical descriptor, the local binary pattern (LBP) histogram [2] has been successfully used for rotation invariant texture classification. Some variants of LBP [3, 4, 5] have been proposed to improve the accuracy and robustness of LBP operator. Filtering is also a pupular method for texture

*Corresponding author

Email addresses: csjxie@gmail.com (Jin Xie), cslzhang@comp.polyu.edu.hk (Lei Zhang), csyjia@comp.polyu.edu.hk (Jane You), csckshiu@comp.polyu.edu.hk (Simon Shiu)

feature extraction. Filter banks [6, 7, 32, 33] such as Gabor filter and wavelet packet, which are good multi-resolution analysis tools, have been utilized to analyze textures. The statistics (e.g., mean and variance) of filtering outputs in each sub-band are used to represent texture images for classification. Translation and rotation invariance can be obtained by some proper filter design techniques; for example, in [7] multi-channel Gabor filters were used to extract rotation invariant texture features.

Another class of texture classification techniques can be categorized as the texton learning based methods [11, 12, 13, 14, 30, 15, 16, 17]. Leung and Malik [11] proposed to build a vocabulary of micro-structures, i.e., textons, for 3D texture classification. For each class, the training images are registered and filtered to produce a set of 48-dimensional response vectors. Then the feature vectors are clustered by using the K-means clustering method, and the resulting cluster centers are called 3D texons. For a test image, its filter response vectors are labeled by the learned textons which are closest to them, and the texton histogram is used as the texture model for classification. Cula and Dana [12] extended Leung and Malik's algorithm to 2D texton learning for un-calibrated and single texture image classification. The methodology is similar to Leung and Malik's except that the occurrences of 3D textons are replaced by 2D textons. Since the dimension of the texton histogram is high, Cula and Dana employed principal component analysis (PCA) to reduce the histogram dimensionality; however, the dimensionality reduced model is much less effective.

Varma and Zisserman [35, 13] modeled textures as their distributions over a set of textons, which are learned from their responses to the MR8 filter bank. The MR8 filter bank is a rotationally invariant, nonlinear filter bank with 38 filters but only 8 filter responses, with which the texton clustering can be conducted in an 8-dimensional space. In [36, 14], textons are clustered directly from the patches of original images instead of their MR8 filter responses. Experimental results showed that the image patch feature can lead to slightly better results than the MR8 feature in terms of classification accuracy. In order to deal with large scale and viewpoint changes of texture images, Varma and Garg [30] extracted the local fractal dimension and length feature from the MR8 filter responses to learn textons for classification.

Lazebnik *et al.* [15] detected the invariant regions in the texture image for texton learning. First, the interest points in the texture image are detected by using the Harris-affine, Hessian-affine and Laplacian of Gaussian detectors. The surrounding region of the detected interest point is normalized by its characteristic scale and main orientation to form the affine invariant region. A combination of spin image (SPIN) and rotation invariant feature transform (RIFT) descriptors is then used to build the texture descriptor on these detected regions. Textons are learned from these descriptors by the K-means clustering method and the texton histogram is built as the statistical feature. Finally, the Earth Mover's Distance (EMD) [31] is used to match the histograms for classification. Based on Lazebnik *et al.*'s work, Zhang *et al.* [16] combined three kinds of descriptors, SPIN, RIFT and scale invariant feature transform (SIFT) [29], to learn textons and employed a kernel SVM classifier for classification.

Recently the sparse representation (or sparse coding) technique [18, 19] has been widely applied to image processing and computer vision tasks [20, 21, 22, 23]. The sparse frame based representation method [37]

and the discriminative dictionary learning method [38] have also been proposed for texture classification. Based on the theory of compressed sensing [40], Liu *et al.* [17] proposed to couple random projection [41] with texton learning for texture classification. Sorted pixels and pixel differences in a patch are projected into a low dimensional space with random projection, and then textons are learned with the K-means clustering method in the compressed domain. The texture description by the learned textons is the same as that in Varma and Zisserman's method [13, 14].

In the above mentioned texton learning based methods, a dictionary of textons is usually learned by the K-means clustering algorithm or the sparse coding algorithm, and the distributions of textons or the histograms of texton encoding coefficients are computed as the statistical features for classification. By the K-means clustering algorithm, however, the learned ball-like clusters may not be able to characterize accurately the intricate feature space of texture images. For example, for a feature vector which lies in the boundary of two or more clusters, the K-means clustering will randomly assign it to one of the neighboring clusters. For the sparse coding based texton learning and encoding algorithms, although some fast sparse coding methods [25, 26] have been proposed, the l_0-norm or l_1-norm minimization still makes these methods time-consuming in order to obtain good accuracy. Moreover, most of these texton learning based texture classification methods [13, 14, 17, 24] use the histogram of texton encoding coefficients for classification, and they will become less effective when the number of training samples is insufficient.

In this paper, a novel texton learning and encoding approach is developed for effective and efficient texture classification. The main motivations and contributions of this work are as follows. 1) Considering that the texture patterns from the same class of texture images are similar, a regularized least square based texton learning method is developed to learn the texton dictionary class by class, and the whole dictionary is concatenated by all sub-dictionaries. 2) A fast two-step l_2-norm texton encoding method is proposed to efficiently code the texture feature over the dictionary. 3) Two types of texton encoding induced histogram features are computed from the coding coefficients and coding residuals, respectively, and they are combined for texture classification via the nearest subspace classifier. The proposed scheme is verified on the representative and benchmark CUReT, KTH_TIPS and UIUC datasets, showing very promising performance, especially when the number of training samples is limited.

The rest of the paper is organized as follows. Section 2 presents in detail the proposed texton learning and encoding scheme for texture classification. Section 3 performs extensive experiments and Section 4 concludes the paper.

2. Texton Encoding Induced Statistical Features for Texture Classification

In this section, we describe in detail the proposed texture classification approach, whose framework is illustrated in Fig. 1. There are four major components in the proposed method: texton dictionary learning, texton encoding, feature description and classification.

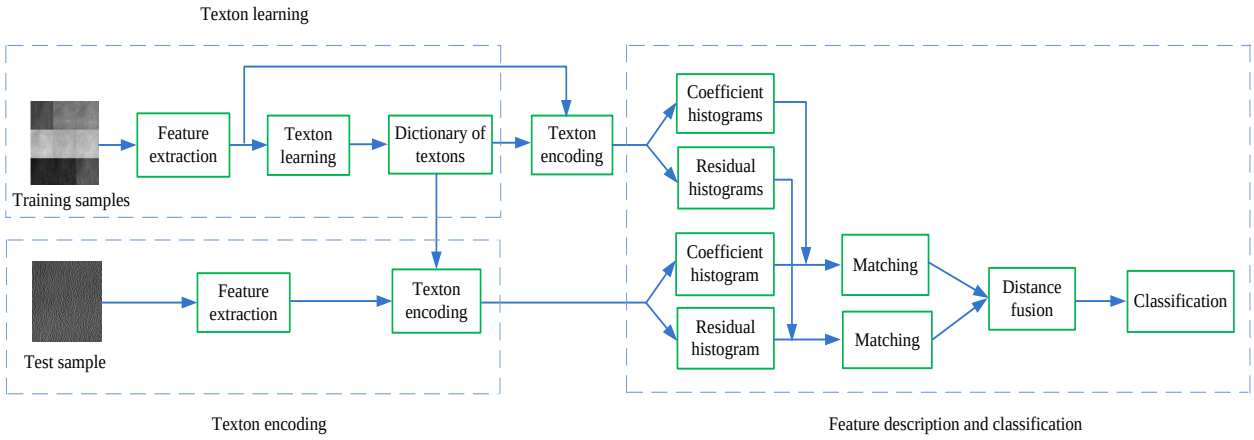

Figure 1: Framework of the proposed texton learning and encoding based texture classification method.

2.1. Texton Dictionary Learning

A dictionary of textons is to be learned from the training texture images so that they can be used to represent the test image. Before learning the textons, the training texture images are first converted into grey level images and are normalized to have zero mean and unit standard deviation. This process reduces the image variations caused by illumination changes. The texture features (e.g., the MR8 feature [13], the patch feature [14] and the SIFT feature [29]) are then extracted from the pre-processed texture images and normalized by using the Weber's law [13]. Suppose that at each pixel of a texture image, we extract an M-dimensional feature vector, denoted by $\boldsymbol{x}_i$, and then for each class of texture images, we can construct a training dataset $\boldsymbol{X} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N] \in \mathcal{R}^{M \times N}$, where N is the number of pixels in all training images of this class. The dictionary of textons, denoted by $\boldsymbol{D} = [\boldsymbol{d}_1, \boldsymbol{d}_2, \ldots, \boldsymbol{d}_L] \in \mathcal{R}^{M \times L}$, is to be learned from the training dataset $\boldsymbol{X}$, where $\boldsymbol{d}_j$, $j = 1, 2, \ldots, L$, is a texton. The dictionary $\boldsymbol{D}$ should be much more compact than $\boldsymbol{X}$, i.e., $L \ll N$, but it is expected that all the samples in $\boldsymbol{X}$ could be well represented by the learned dictionary $\boldsymbol{D}$.

The texture classification methods in [11, 12, 13, 14, 30, 15, 16, 17] use the K-means clustering to learn the textons. Though the K-means clustering is easy and fast to implement, its representation accuracy is limited because only the cluster center is used to approximate the samples. To increase the representation accuracy, we could use the linear combination of several textons, but not the single cluster center, to represent the sample. For example, in [24] the sparse representation model is used to train such a dictionary by solving $min_{\boldsymbol{D},\boldsymbol{\Lambda}}(\|\boldsymbol{X} - \boldsymbol{D}\boldsymbol{\Lambda}\|_F^2 + \lambda\|\boldsymbol{\Lambda}\|_1)$, where λ is a positive constant to balance the representation error and the sparsity of coding coefficients $\boldsymbol{\Lambda}$. However, the sparse coding process is rather time consuming. Furthermore, using sparse representation to learn the dictionary of textons often implies that we have to use sparse representation to encode the query sample, making the texture classification process expensive and slow.

In order to achieve high representation accuracy and efficiency simultaneously, in this paper we propose

the following texton learning model:

$$min_{\boldsymbol{D},\boldsymbol{\Lambda}}(\|\boldsymbol{X}-\boldsymbol{D}\boldsymbol{\Lambda}\|_F^2+\lambda\sum_{i=1}^{N}\|\boldsymbol{\alpha}_i\|_2^2+\gamma\sum_{i=1}^{N}\|\boldsymbol{\alpha}_i-\boldsymbol{\mu}\|_2^2)\ s.t.\ \boldsymbol{d}_j^T\boldsymbol{d}_j=1 \tag{1}$$

where $\boldsymbol{\Lambda}=[\boldsymbol{\alpha}_1,\boldsymbol{\alpha}_2,\ldots,\boldsymbol{\alpha}_N]\in\mathcal{R}^{L\times N}$ is the coding coefficient matrix of $\boldsymbol{X}$ over $\boldsymbol{D}$ and $\boldsymbol{\alpha}_i$, $i=1,2,\ldots,N$, is the L-dimensional coding vector of $\boldsymbol{x}_i$; $\boldsymbol{\mu}$ is the mean of all $\boldsymbol{\alpha}_i$, i.e., $\boldsymbol{\mu}=\frac{1}{N}\sum_{i=1}^{N}\boldsymbol{\alpha}_i$; parameters λ and γ are positive scalars. In general, we require that each texton $\boldsymbol{d}_j$ is a unit vector, i.e., $\boldsymbol{d}_j^T\boldsymbol{d}_j=1$.

In the proposed texton learning model in Eq. (1), we use the l_2-norm, instead of the l_1-norm, to characterize the coding vectors based on the following considerations. First, since the training samples are from the same class, it is not necessary to force the encoding coefficients to be sparse. Second, we learn a compact but not an over-complete dictionary for each class, and l_2-norm regularization is good enough to yield a stable solution to the dictionary. Third, since we will use l_2-norm regularization in the texton encoding stage (please refer to Section 2.2), it is reasonable to also use l_2-norm regularization to learn the dictionary in the training stage. Finally, l_2-norm regularization improves much the speed in training, which is a desirable advantage. In addition, since the training samples $\boldsymbol{x}_i$ from the same class share similarities in appearance, their coding vectors should also be similar. Therefore, in Eq. (1) we enforce their coding vectors $\boldsymbol{\alpha}_i$ to approach to their mean $\boldsymbol{\mu}$, i.e., minimizing $\sum_{i=1}^{N}\|\boldsymbol{\alpha}_i-\boldsymbol{\mu}\|_2^2$. This term is basically to reduce the intra-class variation for more accurate classification.

The optimization of Eq. (1) can be easily conducted by alternatively optimizing $\boldsymbol{D}$ and $\boldsymbol{\Lambda}$. With some random initialization of $\boldsymbol{D}$, we could first fix $\boldsymbol{D}$ and update $\boldsymbol{\Lambda}$, and the problem in Eq. (1) is reduced to a regularized least square problem:

$$\begin{aligned}\hat{\boldsymbol{\alpha}}_i&=argmin_{\boldsymbol{\Lambda}}(\|\boldsymbol{X}-\boldsymbol{D}\boldsymbol{\Lambda}\|_F^2+\lambda\sum_{i=1}^{N}\|\boldsymbol{\alpha}_i\|_2^2+\gamma\sum_{i=1}^{N}\|\boldsymbol{\alpha}_i-\boldsymbol{\mu}\|_2^2)\\&=argmin_{\boldsymbol{\Lambda}}(\|\boldsymbol{X}-\boldsymbol{D}\boldsymbol{\Lambda}\|_F^2+\lambda\sum_{i=1}^{N}\|\boldsymbol{\alpha}_i\|_2^2+\gamma(\sum_{i=1}^{N}\|\boldsymbol{\alpha}_i\|_2^2-N\|\boldsymbol{\mu}\|_2^2)).\end{aligned} \tag{2}$$

Let the partial derivative of Eq. (2) with respect to $\boldsymbol{\alpha}_i$ equal to 0, we have:

$$(\boldsymbol{D}^T\boldsymbol{D}+\lambda\boldsymbol{I}+\gamma\boldsymbol{I})\boldsymbol{\alpha}_i-\boldsymbol{D}^T\boldsymbol{x_i}=\frac{\gamma}{N}\sum_{i=1}^{N}\boldsymbol{\alpha}_i. \tag{3}$$

Summing up Eq. (3) from $i=1$ to $i=N$, we have:

$$\sum_{i=1}^{N}\boldsymbol{\alpha}_i=(\boldsymbol{D}^T\boldsymbol{D}+\lambda\boldsymbol{I})^{-1}\boldsymbol{D}^T\sum_{i=1}^{N}\boldsymbol{x_i}. \tag{4}$$

It is easy to derive that each coding vector $\boldsymbol{\alpha}_i$ in $\boldsymbol{\Lambda}$ can be analytically solved by:

$$\hat{\boldsymbol{\alpha}}_i=(\boldsymbol{D}^T\boldsymbol{D}+\lambda\boldsymbol{I}+\gamma\boldsymbol{I})^{-1}(\boldsymbol{D}^T\boldsymbol{x_i}+\frac{\gamma}{N}(\boldsymbol{D}^T\boldsymbol{D}+\lambda\boldsymbol{I})^{-1}\boldsymbol{D}^T\sum_{i=1}^{N}\boldsymbol{x}_i). \tag{5}$$

Once all the coding vectors are updated, we could fix $\boldsymbol{\Lambda}$ and update $\boldsymbol{D}$. The objective function in Eq. (1) is then reduced to $min_{\boldsymbol{D}}\|\boldsymbol{X}-\boldsymbol{D}\boldsymbol{\Lambda}\|_F^2\ s.t.\ \boldsymbol{d}_j^T\boldsymbol{d}_j=1$. We update the textons $\boldsymbol{d}_j$ one by one. When updating $\boldsymbol{d}_j$, the other textons $\boldsymbol{d}_k(k\neq j)$ are fixed. Denote by $\boldsymbol{\beta}_j$ the j^{th} row of $\boldsymbol{\Lambda}$. We have

$$\hat{\boldsymbol{d}}_j=argmin_{\boldsymbol{d}_j}\|\boldsymbol{X}-\sum\nolimits_{k\neq j}\boldsymbol{d}_k\boldsymbol{\beta}_k-\boldsymbol{d}_j\boldsymbol{\beta}_j\|_F^2\ s.t.\ \boldsymbol{d}_j^T\boldsymbol{d}_j=1. \tag{6}$$

Let $\boldsymbol{Z} = \boldsymbol{X} - \sum_{k \neq j} \boldsymbol{d}_k \boldsymbol{\beta}_k$, and then the above equation can be re-written as:

$$\hat{\boldsymbol{d}}_j = argmin_{\boldsymbol{d}_j} \|\boldsymbol{Z} - \boldsymbol{d}_j \boldsymbol{\beta}_j\|_F^2 \ s.t. \ \boldsymbol{d}_j^T \boldsymbol{d}_j = 1. \tag{7}$$

By using the Lagrange multiplier, we can obtain:

$$\hat{\boldsymbol{d}}_j = argmin_{\boldsymbol{d}_j} tr((\boldsymbol{Z} - \boldsymbol{d}_j \boldsymbol{\beta}_j)(\boldsymbol{Z} - \boldsymbol{d}_j \boldsymbol{\beta}_j)^T - \gamma \boldsymbol{d}_j \boldsymbol{d}_j^T - \gamma). \tag{8}$$

Let the partial derivative of Eq. (8) with respect to $\boldsymbol{d}_j$ equal to 0, we have:

$$-\boldsymbol{Z}\boldsymbol{\beta}_j^T + \boldsymbol{d}_j(\boldsymbol{\beta}_j \boldsymbol{\beta}_j^T - \gamma) = 0. \tag{9}$$

It can be easily derived that the solution to Eq. (8) is

$$\hat{\boldsymbol{d}}_j = \boldsymbol{Z}\boldsymbol{\beta}_j^T / \|\boldsymbol{Z}\boldsymbol{\beta}_j^T\|_2. \tag{10}$$

Once all the textons $\boldsymbol{d}_j$ in $\boldsymbol{D}$ are updated, we then fix $\boldsymbol{D}$ and update the coding coefficients $\boldsymbol{\Lambda}$ by using Eq. (5), and in turn fix $\boldsymbol{\Lambda}$ and update the dictionary $\boldsymbol{D}$ by Eq. (10). Since in each iteration the energy function in Eq. (1) can only reduce, the alternative optimization process will converge to a local minimum, and the desired texton dictionary $\boldsymbol{D}$ is learned.

2.2. Texton Encoding

By using the texton learning algorithm presented in Section 2.1, for each class k, $k = 1, 2, \ldots, C$, we can learn a texton dictionary $\boldsymbol{D}_k$. Then the intra-class dictionaries learned from all the C classes can be concatenated into a dictionary $\boldsymbol{\Phi} = [\boldsymbol{D}_1, \boldsymbol{D}_2, \ldots, \boldsymbol{D}_C]$ to encode the input texture feature vectors for classification. In the K-means clustering based texton learning and encoding methods [11, 12, 13, 14, 15, 16, 17], each texture feature vector is encoded as the label of the texton closest to it. A histogram is built by counting the frequencies of texton labels and it is consequently used as the statistical feature to describe the texture image.

The texton learning and encoding in the K-means clustering based methods [13, 14] is simple but rather coarse. The proposed texton learning model in Eq. (1) can have much higher representation accuracy than the K-means clustering method because it uses a few textons to represent the feature vector. A corresponding texton encoding model needs to be proposed to encode the test feature vector. Let us denote by $\boldsymbol{y}_i$ the feature vector extracted at position i of a given texture image. One may encode $\boldsymbol{y}_i$ by $\hat{\boldsymbol{\alpha}}_i = argmin_{\boldsymbol{\alpha}_i}(\|\boldsymbol{y}_i - \boldsymbol{\Phi}\boldsymbol{\alpha}_i\|_2^2 + \lambda\|\boldsymbol{\alpha}_i\|_2^2)$, which is very fast to optimize because $\hat{\boldsymbol{\alpha}}_i = \boldsymbol{P} \cdot \boldsymbol{y}_i$ and $\boldsymbol{P} = (\boldsymbol{\Phi}^T\boldsymbol{\Phi} + \lambda\boldsymbol{I})^{-1}\boldsymbol{\Phi}^T$ can be pre-calculated as a projection matrix. Different from the class-by-class dictionary learning model in Eq. (1), where the goal is to improve the representation power of $\boldsymbol{D}$ for a given class, here $\boldsymbol{\Phi}$ is the concatenated dictionary of all classes and our goal is to have a discriminative encoding of $\boldsymbol{y}_i$ for accurate classification. Using l_2-norm to regularize $\boldsymbol{\alpha}_i$ is not effective to achieve this goal since l_2-norm tends to generate many big coefficients over different classes. Intuitively, employing l_1-norm regularization in the encoding is able to generate sparse and more discriminative coding coefficients:

$\hat{\boldsymbol{\alpha}}_i = argmin_{\boldsymbol{\alpha}_i}(\|\boldsymbol{y}_i - \boldsymbol{\Phi}\boldsymbol{\alpha}_i\|_2^2 + \lambda\|\boldsymbol{\alpha}_i\|_1)$. However, the l_1-minimization is very time consuming, particularly for the concatenated big dictionary $\boldsymbol{\Phi}$. Though many fast l_1-minimization techniques have been proposed, such as FISTA [25] and ALM [26], it is still highly desirable if we could find an l_2-minimization based texton encoding method while preserving a certain degree of sparsity. We propose a two-step encoding scheme to achieve this goal.

First, we project $\boldsymbol{y}_i$ by using the pre-calculated projection $\boldsymbol{P}$: $\hat{\boldsymbol{\alpha}}_i = \boldsymbol{P} \cdot \boldsymbol{y}_i$, and select the p most related textons to $\boldsymbol{y}_i$ from $\boldsymbol{\Phi}$ by identifying the p most significant coding coefficients in $|\hat{\boldsymbol{\alpha}}_i|$. Usually we set $p \leq M$, where M is the dimension of the feature vector $\boldsymbol{y}_i$. Denote by $\boldsymbol{d}_{i,1}, \boldsymbol{d}_{i,2}, \ldots, \boldsymbol{d}_{i,p}$ the selected p textons to $\boldsymbol{y}_i$. We can thus form a small but adaptive sub-dictionary to $\boldsymbol{y}_i$ as $\boldsymbol{\Phi}_i = [\boldsymbol{d}_{i,1}, \boldsymbol{d}_{i,2}, \ldots, \boldsymbol{d}_{i,p}]$.

We then encode $\boldsymbol{y}_i$ over the sub-dictionary $\boldsymbol{\Phi}_i$ by

$$\hat{\boldsymbol{\theta}}_i = argmin_{\boldsymbol{\theta}_i}(\|\boldsymbol{y}_i - \boldsymbol{\Phi}_i\boldsymbol{\theta}_i\|_2^2 + \lambda\|\boldsymbol{\theta}_i\|_2^2). \tag{11}$$

Clearly, this is a simple regularized least square problem as we have $\hat{\boldsymbol{\theta}}_i = \boldsymbol{P}_i \cdot \boldsymbol{y}_i$ and $\boldsymbol{P}_i = (\boldsymbol{\Phi}_i^T\boldsymbol{\Phi}_i + \lambda\boldsymbol{I})^{-1}\boldsymbol{\Phi}_i^T$. Since p ($p \leq M$) is generally small (please refer to Section 3.4 for more information), the computation of $\boldsymbol{\theta}_i$ is very fast. We can also use the conjugate gradient method [39] to further speed up this encoding process.

The proposed two-step fast texton encoding scheme can be roughly viewed as a hybrid l_0-l_2-minimization with $\|\boldsymbol{\theta}_i\|_0 \leq p$. The first step selects the p textons and actually sets the coding coefficients over the other textons as zeros. This leads to a sparse representation of $\boldsymbol{y}_i$. The second step refines the coding coefficients over the selected p textons to make the representation accurate.

2.3. Feature Description

By using the proposed texton encoding scheme, we could naturally define two types of statistical histogram features to describe the texture image. The first feature descriptor comes from the encoding coefficient $\hat{\boldsymbol{\theta}}_i$. We normalize $\hat{\boldsymbol{\theta}}_i$ into $\boldsymbol{\omega}_i$ by

$$\omega_{i,t} = |\hat{\theta}_{i,t}| / \sum\nolimits_{q=1}^{p} |\hat{\theta}_{i,q}|, t = 1, 2, \ldots, p \tag{12}$$

where $\omega_{i,t}$ is the t^{th} element of $\boldsymbol{\omega}_i$ and $\hat{\theta}_{i,t}$ is the t^{th} element of $\hat{\boldsymbol{\theta}}_i$. Thus, for each feature vector $\boldsymbol{y}_i$ of the texture image, we have a representation vector $\boldsymbol{v}_i$, where only the entries corresponding to the same textons as those in $\boldsymbol{\Phi}_i$ will have non-zero values, i.e., $\omega_{i,t}$, while the other entries in $\boldsymbol{v}_i$ are zeros. Then we can form a histogram, denoted by $\boldsymbol{h}_c$, as one statistical descriptor of this texture image:

$$\boldsymbol{h}_c = \sum\nolimits_{i=1}^{n} \boldsymbol{v}_i \tag{13}$$

where n is the number of pixels in the texture image.

Apart from the encoding coefficient induced histogram, we can also build another histogram by using the encoding residual associated with each of the selected p textons:

$$e_{i,t} = \|\boldsymbol{y}_i - \boldsymbol{d}_{i,t}\hat{\theta}_{i,t}\|_2, t = 1, 2, \ldots, p \tag{14}$$

where $e_{i,t}$ is the t^{th} element of $\boldsymbol{e}_i$. Since the smaller the residual, the more important the associated texton in representing the feature vector, we normalize $\boldsymbol{e}_i$ into $\boldsymbol{\delta}_i$ by

$$\delta_{i,t} = \frac{1/e_{i,t}}{\sum_{q=1}^{p} 1/e_{i,q}}, t = 1, 2, \ldots, p \tag{15}$$

where $\boldsymbol{\delta}_i$ is the normalized reciprocal of encoding residual associated with $\boldsymbol{\Phi}_i$ and $\delta_{i,t}$ is the t^{th} element of $\boldsymbol{\delta}_i$. With $\boldsymbol{\delta}_i$ we can readily re-construct the encoding residual vector of $\boldsymbol{y}_i$ over $\boldsymbol{\Phi}$, denoted by $\boldsymbol{s}_i$. That is, in $\boldsymbol{s}_i$ the entries corresponding to the same textons as those in $\boldsymbol{\Phi}_i$ will have the same values as in $\boldsymbol{\delta}_i$, and the remaining entries in $\boldsymbol{s}_i$ are zeros. Finally, for each feature vector $\boldsymbol{y}_i$ of the texture image, we have an encoding residual induced vector $\boldsymbol{s}_i$, and then we can form a histogram, denoted by $\boldsymbol{h}_r$, as another statistical descriptor of the texture image:

$$\boldsymbol{h}_r = \sum_{i=1}^{n} \boldsymbol{s}_i. \tag{16}$$

Histograms $\boldsymbol{h}_c$ and $\boldsymbol{h}_r$ are two statistical descriptors induced by the encoding coefficients and encoding residuals, respectively. Fig. 2 shows two texture images of different classes and their histograms $\boldsymbol{h}_c$ and $\boldsymbol{h}_r$. We can see that the histogram features of different classes are very different. Meanwhile, the two types of histograms of the same texture image, $\boldsymbol{h}_c$ and $\boldsymbol{h}_r$, also have enough difference, implying that they have complementary information for texture classification.

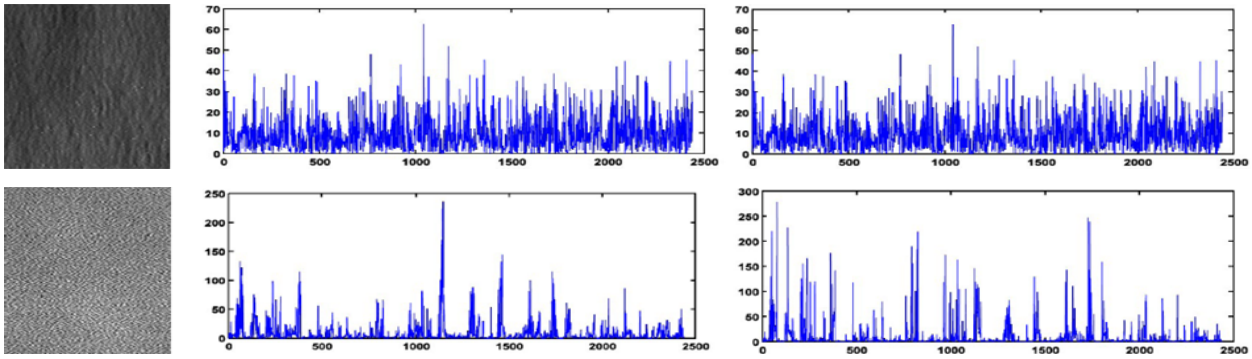

Figure 2: Two texture images (the left column) and their histogram features $\boldsymbol{h}_c$ and $\boldsymbol{h}_r$ (the right two columns).

2.4. Classification

Suppose there are C classes of textures in the dataset, and there are l training samples per class. Denote by $\boldsymbol{H}_c = [\boldsymbol{h}_{c,1}, \boldsymbol{h}_{c,2}, \ldots, \boldsymbol{h}_{c,l}]$ and $\boldsymbol{H}_r = [\boldsymbol{h}_{r,1}, \boldsymbol{h}_{r,2}, \ldots, \boldsymbol{h}_{r,l}]$ the sets of encoding coefficient histograms and encoding residual histograms for one class, respectively. For a test texture image $\boldsymbol{y}$, we first build its two histogram descriptors, denoted by $\boldsymbol{h}_c^y$ and $\boldsymbol{h}_r^y$, and then we use the nearest subspace classifier (NSC) to classify $\boldsymbol{y}$. We project $\boldsymbol{h}_c^y$ and $\boldsymbol{h}_r^y$ into the subspaces spanned by $\boldsymbol{H}_c$ and $\boldsymbol{H}_r$, respectively, as follows:

$$\boldsymbol{\rho}_c = (\boldsymbol{H}_c^T \boldsymbol{H}_c)^{-1} \boldsymbol{H}_c^T \boldsymbol{h}_c^y; \boldsymbol{\rho}_r = (\boldsymbol{H}_r^T \boldsymbol{H}_r)^{-1} \boldsymbol{H}_r^T \boldsymbol{h}_r^y \tag{17}$$

The projection residuals can be computed as:

$$err_c = \|\boldsymbol{h}_c^y - \boldsymbol{H}_c \boldsymbol{\rho}_c\|_2; err_r = \|\boldsymbol{h}_r^y - \boldsymbol{H}_r \boldsymbol{\rho}_r\|_2. \tag{18}$$

Using Eqs. (17) and (18), for each of the C classes, we can calculate the residuals, and then form two vectors $\boldsymbol{err}_c$ and $\boldsymbol{err}_r$ which contain the projection residuals from all classes. By NSC, with either $\boldsymbol{err}_c$ or $\boldsymbol{err}_r$, we can classify $\boldsymbol{y}$ by either $identity(\boldsymbol{y}) = argmin_k \boldsymbol{err}_c(k)$ or $identity(\boldsymbol{y}) = argmin_k \boldsymbol{err}_r(k), k = 1, 2, \ldots, C$. To exploit the discriminative information from both coding coefficients and coding residuals, we fuse $\boldsymbol{err}_c$ and $\boldsymbol{err}_r$ for classification. Here we adopt the simple weighted average method for fusion:

$$\boldsymbol{err}_f = w \cdot \boldsymbol{err}_c + (1 - w) \cdot \boldsymbol{err}_r, 0 \leq w \leq 1 \tag{19}$$

where w is the weight and it can be trained from the training dataset with the "leave-one-out" strategy. Then the classification can be done via

$$identity(\boldsymbol{y}) = argmin_k \boldsymbol{err}_f(k), k = 1, 2, \ldots, C. \tag{20}$$

3. Experimental Evaluation

3.1. Texture Datasets

To demonstrate the effectiveness of our proposed texture classification method, three representative and benchmark texture datasets are used in our experiments: CUReT [27], KTH_TIPS [28] and UIUC [15]. In accordance with other studies which use the CUReT dataset for classification, we use the same subset of CUReT provided by Varma and Zisserman [13], which contains 61 classes and 92 images per class. There are a few factors that make the CUReT dataset challenging. It has both large inter-class similarity and intra-class variation. The images were captured under unknown viewpoint and illumination. Fig. 3 shows some sample images from two different classes in the CUReT dataset. One can see that their appearances are very similar.

A drawback of the CUReT dataset is its lack of large scale variations. Hence, the KTH_TIPS dataset was established to supplement CUReT additional sample images with scale variations. The KTH_TIPS dataset contains 10 classes of materials which are presented in the CUReT dataset. Each class was imaged at 9 different distances while at each distance the images were captured under 3 different directional illuminations and 3 different viewpoints. As a result, 81 texture images are provided for each class. In our experiment, we treat it as a stand-alone dataset, following the setting in [16]. Fig. 4 shows some texture images of two different classes in the KTH_TIPS dataset.

The UIUC dataset contains 25 classes, each having 40 images. A major property of the UIUC dataset is that there are some significant scale and viewpoint changes as well as some non-rigid deformations in its images. Although the UIUC dataset has less severe lighting variations than the CUReT dataset, in terms of intra-class variations of appearance, it is the most challenging one among the commonly used datasets for texture classification. Fig. 5 shows some texture images in the UIUC dataset.

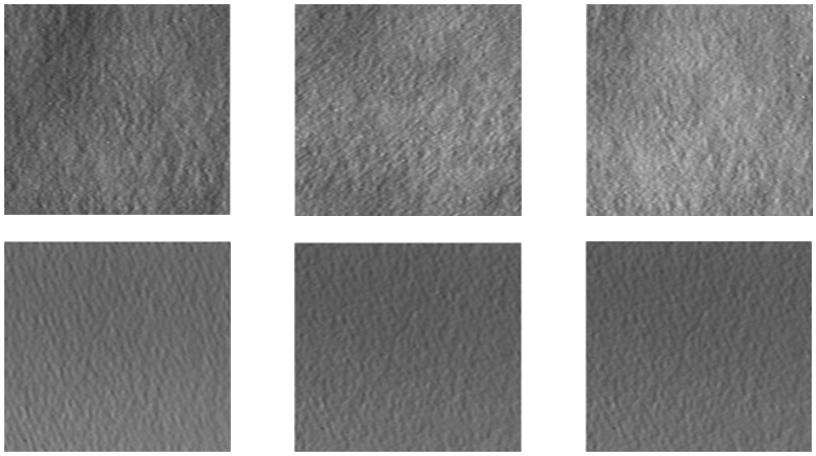

Figure 3: Each row shows three sample images of one texture class in the CUReT dataset.

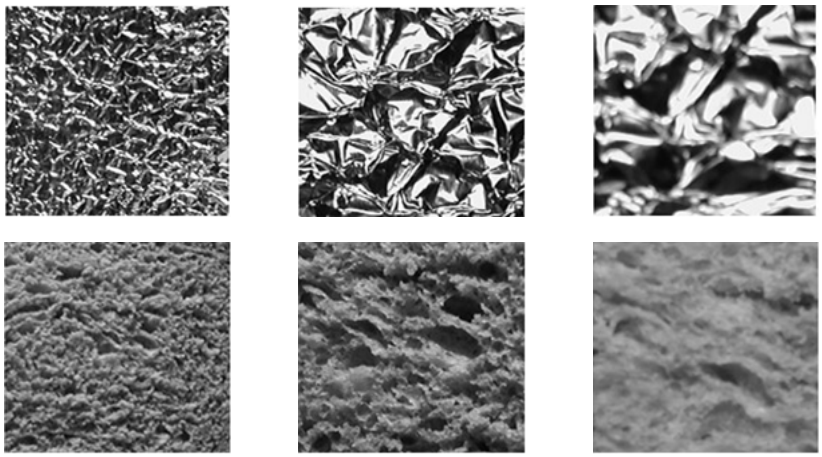

Figure 4: Each row shows three sample images of one texture class in the KTH_TIPS dataset, which has large scale variations.

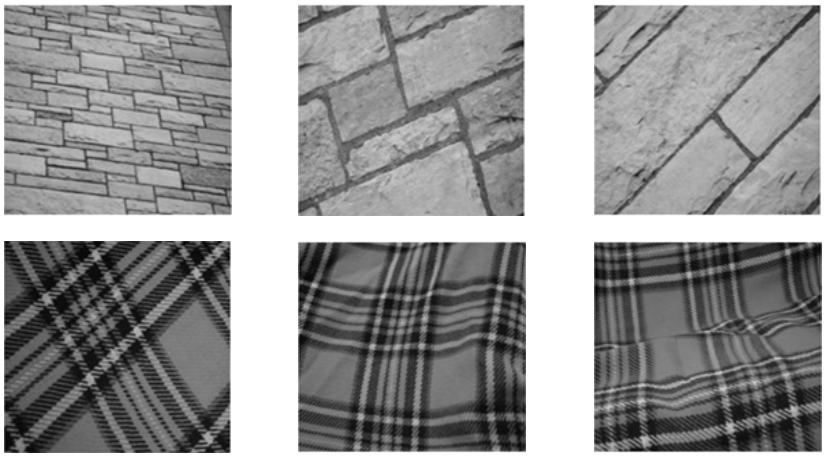

Figure 5: Each row shows three sample images of one texture class in the UIUC dataset, which has large viewpoint changes and non-rigid deformations.

3.2. Parameter Selection and Implementation Details

The evaluation methodology on the three datasets is as follows: l images are randomly chosen per class for training and the remaining images are used for testing. On CUReT, 6, 12, 23 and 46 images per class are randomly chosen to form the training set; on KTH_TIPS, 5, 10, 20, 30 and 40 images per class are randomly chosen to form the training set; on UIUC, 5, 10, 15 and 20 training images are randomly chosen per class to form the training set. For each setting, the experiments were run 100 times and the average classification accuracy is reported. In texton dictionary learning of our model, parameters λ and γ in Eq. (1) are set to 2×10^{-4}. In texton encoding with the learned dictionary, parameter λ is also set to 2×10^{-4}. These parameters are fixed on all the three texture datasets.

On the CUReT and KTH_TIPS datasets, we employed the MR8 feature [13] to represent the texture image for texton learning and encoding. For each class, $L = 40$ textons are learned. The MR8 feature vector is 8-dimensional, i.e., $M = 8$, and thus in the two-step fast texton encoding process we choose $p = 7$ textons to form the sub-dictionary for encoding.

Since the texture images in the UIUC dataset are with large scale and viewpoint variations, using the MR8 feature alone cannot achieve good classification performance. Many existing texture classification methods [15, 16, 17, 34] employ multiple types of features on the UIUC texture dataset. For example, in [15], the SPIN and RIFT descriptors are combined to learn the dictionary for classification, while in [16] the SPIN, RIFT and SIFT descriptors are combined to learn the dictionary. In [17] and [34], the combination of sorted random projection features and the combination of three kinds of multi-fractal spectrums are used to classify the UIUC dataset, respectively. Here we employ the MR8 feature [13] and the SIFT feature [29] to represent the texture images in the UIUC dataset to address their large scale and viewpoint variations. For each class, 100 MR8 textons and 100 SIFT textons per class are learned, respectively. Their histogram features are combined to classify the texture images. For the MR8 feature, $p = 7$ is set in texton encoding. For the 128-dimensional SIFT feature, we set $p = 100$.

For the projection residual fusion in NSC based classification, the weight w is determined by applying the leave-one-out method on the training set. When 6, 12, 23 and 46 training samples per class are used in the experiments on the CUReT dataset, the values of w are 0.5, 0.65, 0.4 and 0.5, respectively. On the KTH_TIPS dataset, when 5, 10, 20, 30 and 40 training samples per class are used, the weights are 0.65, 0.95, 0.65, 0.75 and 0.65, respectively. On the UIUC dataset, when 5, 10, 15 and 20 randomly chosen training samples are used, the weights are 0.6, 0.75, 0.75 and 0.55, respectively.

3.3. Experimental Results

Evaluation of the proposed method: We denote by TEISF_c, TEISF_r and TEISF_f the proposed texton encoding induced statistical feature (TEISF) based methods with only the coding coefficient histogram feature, with only the coding residual histogram feature, and with the fused feature, respectively. We compare the performance of TEISF_c, TEISF_r and TEISF_f on the three texture datasets with different

numbers of training samples. The experimental results on CUReT, KTH_TIPS and UIUC are listed in Tables 1, 2 and 3, respectively. Clearly, compared with TEISF_c and TEISF_r, the classification accuracies with TEISF_f are much improved. Moreover, when the number of training samples is small, TEISF_c and TEISF_r may not get very promising results; however, the fusion of them, i.e., TEISF_f, works very well. This implies that the encoding coefficients and encoding residuals have complementary information for classification. For example, in the experiment where 6 training samples were randomly chosen from each class on the CUReT dataset, 186 texture images are miss-classified by TEISF_f while 647 and 539 images are incorrectly classified with TEISF_c and TEISF_r, respectively. Fig. 6 illustrates some example images from the 3 most frequently mis-classified classes by TEISF_f.

Table 1: Classification rates on CUReT by TEISF_c, TEISF_r and TEISF_f.

Training samples	6	12	23	46
TEISF_c	87.19%	93.4%	97.29%	99.03%
TEISF_r	88.35%	93.83%	97.43%	99.15%
TEISF_f	**95.21%**	**98.5%**	**99.25%**	**99.54%**

Table 2: Classification rates on KTH_TIPS by TEISF_c, TEISF_r and TEISF_f.

Training samples	5	10	20	30	40
TEISF_c	92.11%	95.61%	96.99%	97.06%	97.53%
TEISF_r	92.16%	95.18%	96.70%	97.30%	97.67%
TEISF_f	**92.95%**	**95.77%**	**98.1%**	**98.3%**	**98.9%**

Table 3: Classification rates on UIUC by TEISF_c, TEISF_r and TEISF_f.

Training samples	5	10	15	20
TEISF_c	91.05%	95.30%	97.38%	98.18%
TEISF_r	93.29%	95.82%	97.65%	98.22%
TEISF_f	**94.37%**	**98.38%**	**99.35%**	**99.54%**

Comparison evaluation: We compare our methods with representative and recently developed texture classification methods [2, 4, 9, 10, 13, 14, 30, 28, 15, 16, 17]. Since many competing methods do not have publically released source codes available, we cropped the results from their original papers, or we asked the authors to provide the results. If the classification accuracies of some competing methods are not available, we use symbol '–' to represent the missing results. Different competing methods may use different classifiers, e.g., the nearest neighbor classifier with the χ^2-distance in [2, 4, 13, 14, 30, 34, 9], SVM classifier in [28, 16, 17], etc. For the LBP [2], CLBP [4] and VZ_MR8 [13] methods, we coupled them with the NSC classifier, which gives better classification rates than the nearest neighbor classifier with the χ^2-distance as in the original papers.

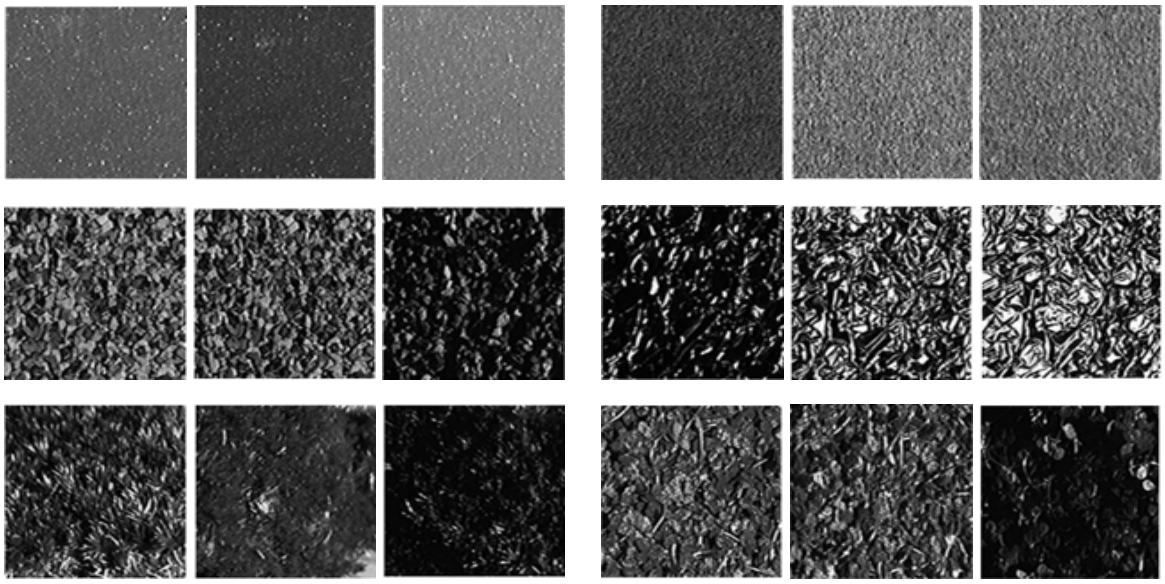

Figure 6: Example images which are mis-classified by the TEISF_f method in the CUReT dataset. The left 3 columns show some test texture images while the right 3 columns show images from the class that these test images are mis-classified to. Specifically, "Styrofoam"(top), "Aquarium stones"(middle) and "Moss"(bottom) are mis-classified to "Cork", "Aluminum foil" and "Soleirolia plant", respectively. From these example images one can see that the mis-classified test images are often perceptually similar to the template images.

Table 4 compares the competing texture classification methods on the three texture datasets when enough training samples are available (46 for CUReT, 40 for KTH_TIPS and 20 for UIUC). One can see that when there are enough training samples, most of the methods can achieve good classification accuracy. The proposed TEISF_f method works the best on the CUReT and UIUC datasets, and it is only slightly worse than Liu *et al.*'s method [17] by a gap of 0.39% on the KTH_TIPS dataset. In this experiment, the gain of TEISF_f over TEISF_c and TEISF_r is not significant because TEISF_c and TEISF_r can also achieve good results when enough training samples are available.

By decreasing the number of training samples per class, in Tables 5, 6 and 7 we present the classification accuracies on the three datasets, respectively. (Note that not all the competing methods employed in Table 4 are reported in Tables 5, 6 and 7 since some results are not available.) From these tables, we can readily make the following findings. 1) The proposed TEISF_f achieves the best classification accuracy in almost every case. 2) With the decrease of the number of training samples, the advantage of TEISF_f over other methods is getting more and more obvious. 3) TEISF_f is much more robust to the number of training samples than other competing methods. For example, on the CUReT dataset, TEISF_f's classification accuracies are 95.21%, 98.5%, 99.25% and 99.54% with 6, 12, 23 and 46 training samples per class, respectively. From 46 training samples per class to 6 training samples per class, the drop in classification rate is only 4.3%. However, for other methods, the drop is often more than 10%. 4) Some methods may work well on one dataset but not so well on other datasets. For example, Xu *et al.*'s method [9, 34] has good accuracy on the UIUC dataset, whereas its accuracy on the CUReT dataset is not so good. In comparison, the proposed

TEISF_f can obtain good results across all datasets. In addition, in Fig. 7 we plot the curves of classification rate vs. number of training samples on the three datasets by TEISF_f. The curves by VZ_MR8 [13]+ NSC, CLBP [4]+NSC and the method in [17] are also plotted for comparison. Clearly, the proposed method is much more robust than other methods to the number of training samples.

The speed of one algorithm in processing one query sample is very important for its practical usage. In the texton encoding stage of our TEISF_f algorithm, the time complexity of selecting the p most related textons is $\mathcal{O}((CL)^2)$, and the complexity of solving the least square problem in Eq. (11) is $\mathcal{O}(p^2M)$, where CL is the number of textons in the concatenated dictionary $\boldsymbol{\Phi}$, p is the number of the selected textons in the texton encoding stage and M is the dimension of the texture feature vector. Hence, the time complexity of texton encoding in the proposed TEISF_f method is $\mathcal{O}(n(CL)^2 + np^2M)$, where n is the number of input feature vectors of a test image. In the feature description stage, since at each pixel in the image there are CL operations and $4Mp + CL$ operations to form the coefficient histogram and residual histogram, respectively, the time complextiy of feature description is $\mathcal{O}(nMp + nCL)$. In the classification stage, the time complextiy of computing the projection residuals and fusing them is $\mathcal{O}(l^2CL)$, where l is the number of training images per class.

Let us then compare the average running time of the proposed TEISF_f method with the K-means clustering based VZ_MR8 method [13] and the l_1-sparsity based texton learning method [24], where a texton encoding procedure is employed to classify the query sample. On the CUReT and KTH_TIPS datasets, 40 textons per class are used in the texton encoding of the proposed method, and on the UIUC dataset, 100 MR8 textons and 100 SIFT textons per class are employed, respectively. All experiments were performed in the Matlab programming environment on a laptop with a 2.30 GHz quad-core Intel processor and 8 GB memory. The average running time is listed in Table 8. One can see that TEISF_f is much faster than the l_1-sparsity based method [24]. Compared with the K-means clustering based method in [13], although the proposed method is slower, it can obtain much higher classification rate (please refer to Tables 5, 6 and 7). Overall, the proposed method can achieve a good trade-off between the classification rate and the computational complexity.

3.4. Effects of Parameters L and p

In this subsection, we study the effects of parameters L and p on the final classification accuracy. Parameter L is the number of textons per class in the dictionary, which controls the capacity of the dictionary to represent the texture appearance. We conduct a series of experiments with different numbers of textons on the three texture datasets. Fig. 8 plots the curves of classification rate v.s. number of leaned textons from each class. One can see that in the proposed TEISF_f method the number of learned textons, which ranges from 20 to 100, has little effect on the final classification accuracy.

The parameter p controls the number of selected textons in the texton encoding stage. In some sense it controls the sparsity of coding coefficients in the texton encoding. Fig. 9 shows the classification rates under different numbers of selected textons on the CUReT, KTH_TIPS and UIUC datasets, respectively. Note that

Table 4: Texture classification rates by different methods when the number of training samples is relatively large.

	CUReT(46)	KTH_TIPS(40)	UIUC(20)
LBP [2]+NSC	99.11%	97.19%	80.3%
CLBP [4]+NSC	96.19%	96.74%	95.18%
VZ_MR8 [13]+NSC	98.17%	97.06%	96.74%
VZ_Patch [14]	98.03%	92.40%	97.83%
Hayman *et al.* [28]	98.46%	94.8%	92%
Lazebnik *et al.* [15]	72.5%	91.13%	93.62%
Zhang *et al.* [16]	95.3%	96.1%	98.7%
Varma and Garg [30]	97.5%	——	95.4%
Crosier and Griffin [10]	98.6%	98.5%	98.8%
Xu *et al.* [9]	——	——	98.60%
Liu *et al.* [17]	99.37%	**99.29%**	98.56%
TEISF_c	99.03%	97.53%	98.18%
TEISF_r	99.05%	97.67%	98.22%
TEISF_f	**99.54%**	98.9%	**99.54%**

Table 5: Classification rates on the CUReT dataset with different numbers of training samples.

Training samples	6	12	23	46
LBP [2]+NSC	82.25%	91.71%	96.24%	99.11%
CLBP [4]+NSC	79.75%	90.51%	94.47%	96.19%
VZ_MR8 [13]+NSC	86.33%	92.79%	96.45%	98.17%
Varma and Garg [30]	81.67%	89.74%	94.69%	97.5%
Liu *et al.* [17]	86.48%	96.43%	97.71%	99.37%
TEISF_c	87.19%	93.4%	97.29%	99.03%
TEISF_r	88.35%	93.83%	97.43%	99.15%
TEISF_f	**95.21%**	**98.5%**	**99.25%**	**99.54%**

two dictionaries (for MR8 and SIFT features) are used on the UIUC dataset. Thus in the right sub-figure of Fig. 9, the x-axis labels a pair of numbers of the selected MR8 and SIFT textons. From these figures, we can observe that when p is large (e.g., $p > 15$ for the MR8 feature on the CUReT dataset), the sparsity of coding coefficients is reduced and the classification rate also decreases. When p is very small (e.g., $p < 5$ for the MR8 feature on the CUReT dataset), the representation is not accurate so that the classification rate is not very good, either. Emperically, when p is slightly less than the feature dimensionality (e.g., $p = 7$ for the 8-dimensional MR8 feature), a good balance of representation accuracy and sparsity is reached so that good classification rate can be obtained. For example, on the CUReT dataset, with 46 training samples per

Table 6: Classification rates on the KTH_TIPS dataset with different numbers of training samples.

Training samples	5	10	20	30	40
LBP [2]+NSC	72.50%	85.41%	94.12%	95.73%	97.19%
CLBP [4]+NSC	78.51%	87.69%	94.78%	96.05%	96.74%
VZ_MR8 [13]+NSC	90.46%	93.45%	95.32%	95.70%	97.06%
Liu *et al.* [17]	80.90%	89.49%	96.40%	**98.4%**	**99.29%**
TEISF_c	92.11%	95.61%	96.99%	97.06%	97.53%
TEISF_r	92.16%	95.18%	96.70%	97.30%	97.67%
TEISF_f	**92.95%**	**95.77%**	**98.1%**	98.3%	98.9%

Table 7: Classification rates on the UIUC dataset with different numbers of training samples.

Training samples	5	10	15	20
LBP [2]+NSC	43.34%	63.15%	74.15%	80.3%
CLBP [4]+NSC	76.73%	89.5%	94%	95.18%
VZ_MR8 [13]+NSC	86.54%	93.56%	95.71%	96.74%
VZ_Patch [14]	90.17%	95.18%	96.94%	97.83%
Lazebnik *et al.* [15]	84.77%	90.17%	92.42%	93.62%
Varma and Garg [30]	85.35%	91.64%	94.09%	95.4%
Xu *et al.* [9]	93.42%	96.95%	98.01%	98.60%
Liu *et al.* [17]	90.96%	96.00%	97.59%	98.56%
TEISF_c	91.05%	95.30%	97.38%	98.18%
TEISF_r	93.29%	95.82%	97.65%	98.22%
TEISF_f	**94.37%**	**98.38%**	**99.35%**	**99.54%**

Table 8: Average running time (second) of the three texton encoding based methods on the CUReT, KTH_TIPS and UIUC texture datasets.

	CUReT	KTH_TIPS	UIUC
VZ_MR8 [13]	7.3s	4.2s	80.9s
l_1-sparsity [24]	133.2s	53.3s	1254.5s
TEISF_f	13.6s	8.0s	217.6s

class, when 7 textons are chosen, a classification rate of 99.54% can be obtained.

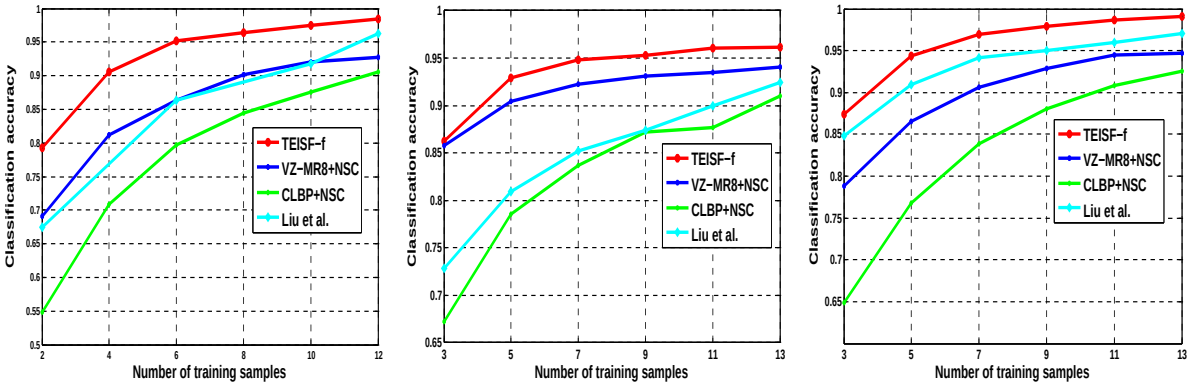

Figure 7: The curves of classification rate vs. number of training samples by TEISF_f, VZ_MR8 [13]+NSC, CLBP [4]+NSC and Liu *et al.*'s method [17] on CUReT (left), KTH_TIPS (middle) and UIUC (right).

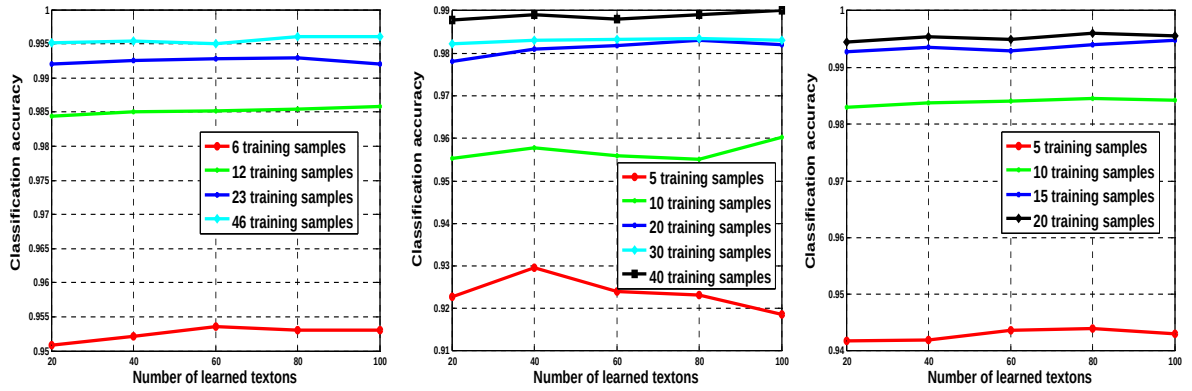

Figure 8: The curves of classification rate vs. number of learned textons from each class of training samples by TEISF_f on CUReT (left), KTH_TIPS (middle) and UIUC (right).

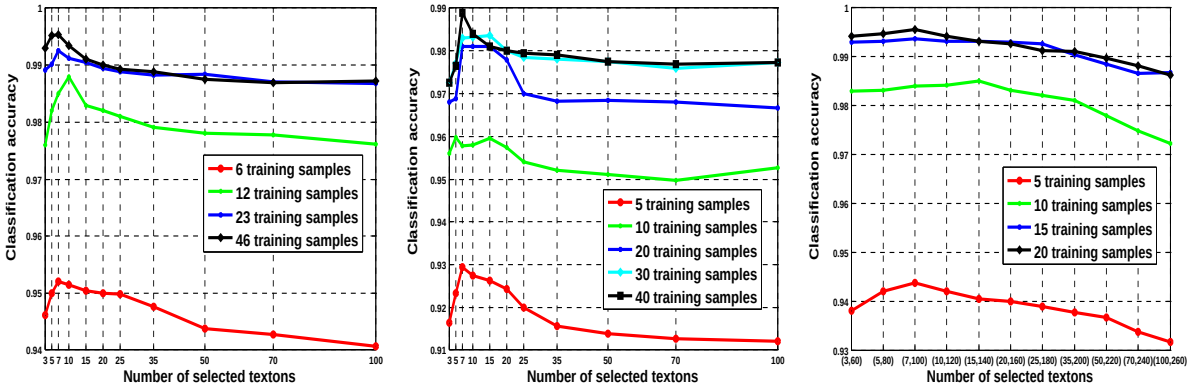

Figure 9: The curves of classification rate vs. number of selected textons in the texton encoding stage by TEISF_f on CUReT (left), KTH_TIPS (middle) and UIUC (right).

4. Conclusions

We proposed a texton encoding based texture classification method, which is simple to implement and shows very promising performance. The textons were learned to ensure the representation accuracy while reducing the within-class variance. A two-step texton encoding scheme was then proposed to encode efficiently the texture feature over the learned dictionary while preserving certain sparsity. Two types of statistical features, namely, coding coefficient induced histogram and coding residual induced histogram, were then defined, with which the nearest subspace classifier was applied for texture classification. The proposed texton encoding induced statistical feature (TEISF) method was validated on three representative and benchmark texture datasets: CUReT, KTH_TIPS and UIUC. The experimental results demonstrated that TEISF achieves very good texture classification performance, especially when the number of training samples is small.

References

[1] R. M. Harlick, K. Shanmugam, I. Dinstein, Texture features for image classification, *IEEE Trans. on Systems, Man, and Cybertics*. 3(6)(1973)610-621.

[2] T. Ojala, M. Pietikainen, T. Maenpaa, Multi-resolution gray-scale and rotation invariant texture classification with local binary patterns, *IEEE Trans. on Pattern Analysis and Machine Intelligence*. 7(24)(2002)971-987.

[3] S. Liao, W. K. Law, C. S. Chung, Dominant local binary pattern for texture classification, *IEEE Trans. on Image Processing*. 18(5)(2009)1107-1118.

[4] Z. H. Guo, L. Zhang, D. Zhang, A completed modeling of local binary pattern operator for texture classification, *IEEE Trans. on Image Processing*. 19(6)(2010) 1657-1663.

[5] Z. H. Guo, L. Zhang, D. Zhang, Rotation invariant texture classification using LBP variance (LBPV) with global matching, *Pattern Recognition*. 43(3)(2010) 706-719.

[6] C. Pun, M. Lee, Log-polar wavelet energy signatures for rotation and scale invariant texture classification, *IEEE Trans. on Pattern Analysis and Machine Intelligence*. 25(5)(2004)1228-1333.

[7] A. C. Bovik, M. Clark, W. S. Geisler, Multichannel texture analysis using localized spatial filters, *IEEE Trans. on Pattern Analysis and Machine Intelligence*. 12(1)(1990)55-73.

[8] M. Mellor, B. W. Hong, M. Brady, Locally roatation, contrast, and scale invariant descriptors for texture analysis, *IEEE Trans. on Pattern Analysis and Machine Intelligence*. 30(1)(2008)52-61.

[9] Y. Xu, X. Yang, H. Ling, H. Ji, A new texture descriptor using multifractal analysis in multi-orientation wavelet pyramid, in *Proc. Computer Vision and Pattern Recognition*, 2010, pp. 161-168.

[10] M. Crosier, L. Griffin, Using basic image features for texture classification, *International Journal of Computer Vision*. 88(3)(2010)447-460.

[11] T. Leung, J. Malik, Representing and recognizing the visual appearance of materials using three-dimensional textons, *International Journal of Computer Vision*. 43(1)(2001)29-44.

[12] O. G. Cula, K. J. Dana, 3D texture recognition using bidirectional feature histograms, *International Journal of Computer Vision*. 59(1)(2004)33-60.

[13] M. Varma, A. Zisserman, A statistical approach to material classification from single images, *International Journal of Computer Vision*. 62(2)(2005)61-81.

[14] M. Varma, A. Zisserman, A statistical approach to material classification using image patch examplars, *IEEE Trans. on Pattern Analysis and Machine Intelligence*. 31(11)(2009)2032-2047.

[15] S. Lazebnik, C. Schmid, J. Ponce, A sparse texture representation using local affine regions, *IEEE Trans. on Pattern Analysis and Machine Intelligence*. 27(8)(2005)1265-1278.

[16] J. Zhang, M. Marszalek, S. Lazebnik, C. Schmid, Local features and kernels for classification of texture and object categories:a comprehensive study. *International Journal of Computer Vision*. 73(2)(2007)213-238.

[17] L. Liu, P. Fieguth, G. Y. Kuang, B. H. Zha, Sorted random projections for robust texture classification, in *Proc. International Conference on Computer Vision*, 2011, pp. 391-398.

[18] D. L. Donoho, M. Elad, Optimal sparse representation in general (nonorthogonal) dictionaries via l_1 minimization, in *the National Academy of Sciences*, 2003, pp. 2197-2202.

[19] A. Bruckstein, D. L. Donoho, M. Elad, From sparse solutions of systems of equations to sparse modeling of signals and images, *SIAM Journal on Review*. 1(51)(2009)34-81.

[20] M. Aharon, M. Elad, A. Bruckstein, The K-SVD: an algorithm for designing of overcomplete dictionaries for sparse representation, *IEEE Trans. on Signal Processing*. 54(11)(2006)4311-4322 .

[21] H. zhang, Y. Zhang, T. Huang, Pose-robust face recognition via sparse representation, *Pattern Recognition*. 46(5)(2013)1511-1521.

[22] Y. Zhou, K. Liu, R. Carrillo, K. Barner, F. Kiamilev, Kernel-based sparse representation for guesture recognition, *Pattern Recognition*. 45(4)(2012)1290-1298.

[23] X. Zhang, Y. Yang, L. Jiao, F. Dong, Manifold-constrained coding and sparse representation for human action recognition, *Pattern Recognition*. 46(7)(2013) 1819-1831.

[24] J. Xie, L. Zhang, J. You, D. Zhang, Texture classification via patch-based sparse texton learning, in *Proc. International Conference Image Processing*, 2010, pp. 2737-2740.

[25] A. Beck, M. Teboulle, A fast iterative shrinkage-thresholding algorithm for linear inverse problems, *SIAM Journal on Imaging Sciences*. 2(1)(2009)183-202.

[26] J. Yang, Y. Zhang, Alternating direction algorithms for l_1-problems in compressive sensing, *SIAM Journal on Scientific Computing*. 33(1)(2011)250-278.

[27] K. Dana, B. VanGinneken, S. K. Nayar, J. J. Koenderink, Reflectance and texture of real world surfaces, *ACM Trans. on Graphics*. 18(1)(1999)1-34.

[28] E. Hayman, B. Caputo, M. Fritz, J. O. Eklundh, On the significance of real-world conditions for material classification, in *Proc. European Conference Computer Vision*, 2004, pp. 253-266.

[29] D. Lowe, Distinctive image features from scale-invariant features, *International Journal of Computer Vision*. 60(2)(2004)91-110.

[30] M. Varma, R. Garg, Locally invariant fractal features for statistical texture classification, in *Proc. International Conference Computer Vision*, 2007, pp.1-8.

[31] E. Levina, P. Bickel, The Earth Mover's distance is the mallows distance: some insights from statistics, in *Proc. International Conference Computer Vision*, 2001, pp.251-256.

[32] M. Do, M. Vetterli, Wavelet-based texture retrieval using generalized Gaussian density and Kullback-Leibler distance, *IEEE Trans. on Image Processing*. 14(10)(2002)146-158.

[33] K. J. Khouzani, H. S. Zaden, Rotation invariant multiresolution texture analysis using radon and wavelet transforms, *IEEE Trans. on Image Processing*. 14(6)(2005)783-794.

[34] Y. Xu, H. Ji, C. Fermuller, Viewpoint invariant texture description using fractal analysis, *International Journal of Computer Vision*. 83(1)(2009)85-100.

[35] M. Varma, A. Zisserman, Classifying images of materials: achieving veiwpoint and illumination independence, in *Proc. European Conference Computer Vision*, 2002, pp. 255-271.

[36] M. Varma, A. Zisserman, Texture classification: are filter banks necessary? in *Proc. Computer Vision and Pattern Recognition*, 2003, pp. 691-698.

[37] K. Skretting, J. H. Husøy, Texture classification using sparse frame based representations, *EURASIP Journal. on Applied Signal Processing*. (2006).

[38] J. Mairal, F. Bach, J. Ponce, G. Sapiro, A. Zisserman, Discriminative learned dictionaries for local image analysis, in *Proc. Computer Vision and Pattern Recognition*, 2008, pp. 1-8.

[39] K. Tanabe, Conjugate-gradient method for computing the Moore-Penrose inverse and rank of a matrix, *Journal of Optimization Theory and Applications*. 22(1)(1977) 1-23.

[40] D. L. Donoho, Compressed sensing, *IEEE Trans. on Information Theory*. 52(4)(2006)1289-1306.

[41] W. Johnson, J. Lindenstrauss, Extensions of Lipschitz mappings into a Hilbert space, in *Proc. Conference in Modern Analysis and Probability*, 1984, pp. 189-206.