The Hong Kong Polytechnic University

Department of Computing

Texton Encoding based Texture Classification and Its Applications to Hand-Back Skin Texture Analysis

by

Jin Xie

A thesis submitted in partial fulfillment of the requirements

for the Degree of Doctor of Philosophy

April 2012

CERTIFICATE OF ORIGINALITY

I hereby declare that this thesis is my own work and that, to the best of my knowledge and belief, it reproduces no material previously published or written, nor material that has been accepted for the award of any other degree of diploma, except where due acknowledgement has been made in the text.

___ ____ (Signed)

__Jin Xie_ (Name of Student)

Abstract

Real world objects have various types of texture surfaces. With the increasing demands of image understanding and object recognition in many computer vision applications, texture classification has been receiving considerable attention, and plenty of texture classification methods have been proposed in the past decades. However, how to efficiently represent texture and extract texture features is still a challenging problem in texture image analysis and classification. In this thesis, we investigate this problem and propose new solutions for efficient texture feature extraction, representation and classification. As an interesting application, we also apply the proposed methods to hand back skin texture analysis.

First, we present a sparse representation (SR) based dictionary learning method to learn a dictionary of textons for texture image representation. In traditional texton learning based texture representation approaches, texton learning is usually implemented by the *K*-means clustering method. However, the *K*-means clustering process may not be able to well characterize the intrinsic feature space of textons, which is often embedded into a low dimensional manifold. To improve the representation accuracy and capability, we propose to use the dictionary learning method under the SR framework to learn a dictionary of textons. Consequently, the SR coefficients of the texture image over the dictionary of textons are used to construct the histograms for classification. The proposed SR based texton dictionary learning method yields better performance than the traditional *K*-means clustering based texture classification methods.

We further propose an efficient texton encoding based texture classification scheme. The scheme consists of four stages: texton dictionary learning, texton encoding, feature description and classification. In the stage of texton dictionary learning, a regularized least square based texton learning model is proposed. Compared with the texton learning based on SR or *K*-means clustering, the proposed model is much more accurate than the *K*-means clustering while being much more efficient than the SR to implement. Meanwhile, we propose a fast texton encoding method to code the texture feature over the learned dictionary. Consequently, two types of texton encoding induced statistical features, coefficient histogram and residual histogram, are extracted for classification. The proposed method, namely texton encoding induced statistical feature (TEISF), is validated on three representative benchmark texture datasets: CUReT, KTH_TIPS and UIUC. The experimental results demonstrate that TEISF outperforms state-of-the-arts, especially when the number of the training samples is small.

Finally, we study the hand back skin texture (HBST) pattern classification problem for personal identification and gender classification. A specially designed HBST imaging system is developed to capture the HBST images, and an HBST image dataset is established, which consists of 1920 images from 80 persons (160 hands). Then the proposed texton learning based texture analysis methods are applied to the established HBST dataset, and the experimental results demonstrate that HBST is very useful to aid human identity identification and gender classification. As a kind of specific texture images, the established HBST dataset is rather challenging and provides a good platform to evaluate various texture classification algorithms.

Publications

The following papers, published or submitted, are the partial outputs of my Ph.D studies at PolyU.

[1] J. Xie, L. Zhang, J. You, and D. Zhang. Texture classification via patch-based sparse texton learning, *ICIP*, 2010，pp. 2737-2740.

[2] J. Xie, L. Zhang, J. You, D. Zhang and X. F. Qu. A Study of Hand Back Skin Texture Pattern for Personal Identification and Gender Classification, *Sensors*, Accepted.

[3] J. Xie, L. Zhang, J. You, D. Zhang and G. P. Zhu. Effective Statistical Features via Texton Encoding for Texture Classification, Submitted to IEEE Trans. Image Processing.

Acknowledgements

This Ph. D study is now going to the end, I’m very glad to take this opportunity to express my appreciation to all the people concerning and supporting me in the past years.

First of all, I’d like to say thank you to my chief supervisor, Dr. Lei Zhang, for his patient and professional supervision on my research work. In the Ph. D study years, Dr. Zhang shows me the scientific idea and unexhausted passion that a researcher should own as well as the endeavor which a researcher should devote to exploiting the unknown field. Dr. Zhang’s diligence and meticulosity really impact me a lot. I will always keep them in my mind in my future work and life.

Besides Dr. Zhang, I’d also like to express my gratitude to other professors, they are Prof. David Zhang, Prof. Jane You, Prof. Qinghua Hu and Dr. Guopu Zhu. They all give me helpful advice during my study. My colleagues also give me great support and kind help in the past Ph.D study, they are Dr. Lin Zhang, Dr. Bo Peng, Dr. Maggie Guo, Dr. Denis Guo, Dr. Jerry Zhao, Dr. Qin Li, Mr. Meng Yang, Mr. Kaihua Zhang, Mr. Pengfei Zhu, Mr. Zhizhao Feng, Mr. Andy Wang, Mr. Xiaofeng Qu, Mr. Jinghua Wang and Miss. Feng Liu. I’d also like to appreciate my friends, without their encourage and help, I would never finish this journey. They are Dr. Lei Ye, Dr. Meng Wang, Dr. Yuan Zhen, Dr. Guobin Liu, Dr. Yong Chen, Miss. Lei Xu, Mr. Duncan Yung.

Lastly, my deepest gratitude goes to my parents and family for their understanding and great mental support, they will not read this thesis, but their support is most critical for me to finish this journey.

Table of Content

List of Figures

List of Tables

Chapter 1. Introduction

The main objective of this thesis is to investigate new solutions for accurate texture image classification. In particular, we focus on the texton learning method to represent texture images and extract effective texture features. With respect to texture representation, we will discuss how to employ dictionary learning techniques to learn textons to represent the texture. With the learned textons, we then extract new histogram features for texture classification. As an application of the proposed texton learning based texture analysis methods, we will investigate the problem of human identification and gender classification with hand back skin texture images. This chapter provides an introduction to the several key issues in texture image analysis. We also summarize the contributions and outline the organization of this thesis.

1.1 Background

Texture analysis is an important research topic in computer vision and pattern recognition, since most real world objects show different kinds of texture surfaces. There are many real world applications that involve texture analysis, including medical imaging, remote sensing and material classification, etc. With the increasing demands of these applications in image understanding and object recognition, texture classification has been receiving considerable attention. The canonical texture classification task is to design an efficient algorithm to categorize previously unseen images to some known class of texture images whose training samples have been provided. In particular, the classification problem mainly depends on how many training samples are available, what properties they have and how they are related to the test texture images to be classified. Usually, the texture classification problem can be viewed as the categorization of textures which are from different materials based on their appearances. Furthermore, no constraints are imposed on the acquisition of the training or testing texture images, and, in particular, no *a priori* knowledge of the illumination, viewpoint and scale change is required when capturing the texture images.

Texture classification has been applied in different fields. In the area of content based image retrieval, texture features are extracted to simultaneously localize and categorize the textures of interest in an image [1]. For example, in Fig. 1.1, we can detect the presence and location of the zebras in the image. The texture pattern of the zebra can be learned by some positive and negative training samples in the top row. The subsequent rows present the top 15 retrieved zebra images from a subset of the COREL dataset by their scores. Moreover, structure texture similarity metrics are employed for texture image retrieval in [2]. With the structural similarity metrics, similar and dissimilar texture pairs can be classified, which is corresponded to human judgments. Fig. 1.2 shows some similar and dissimilar texture pairs according to human judgment scores. Texture similarity metrics are also important for image coding since some similar texture regions in the image do not affect the perceived image quality.

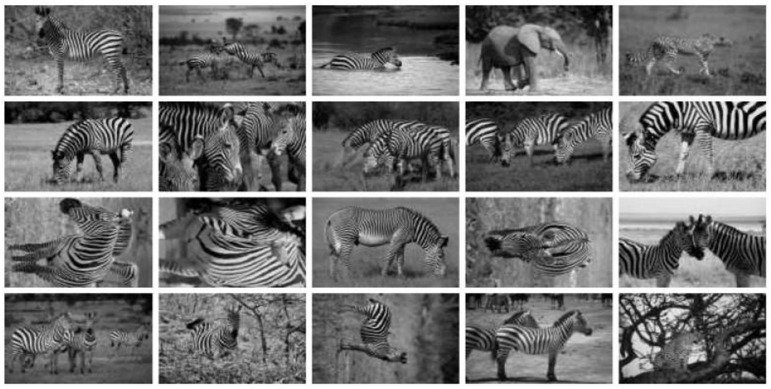


Figure 1.1: Content based image retrieval by using texture features [1]. The texture pattern of the zebra is learned by some positive and negative training samples in the top row. The following three rows show the top 15 retrieved zebra images from a subset of the COREL dataset by their scores.

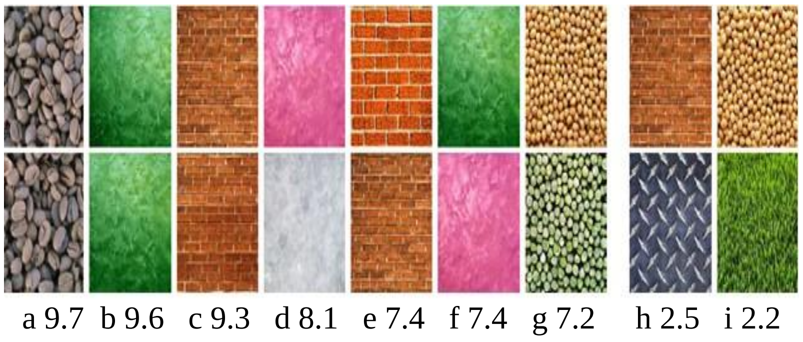


Figure 1.2: Selected texture pairs and their similarity judged by average human scores [2].

Furthermore, texture classification has been extended to applications in medical image analysis. It has been used to screen women for early signs of breast cancer by classifying parenchymal density and detecting microcalcifications [8, 9, 10]. In dermatology, some methods based on skin texture model are developed for computer-assisted diagnosis of skin disorders [11]. In these methods, bidirectional texture function is used to describe the texture surface. Fig. 1.3 illustrates five dermatological disorders which can be classified with the skin texture model: acne, congenital melanocytic nevus (medium sized), keratosis pilaris, psoriasis, and acute allergic contact dermatitis.

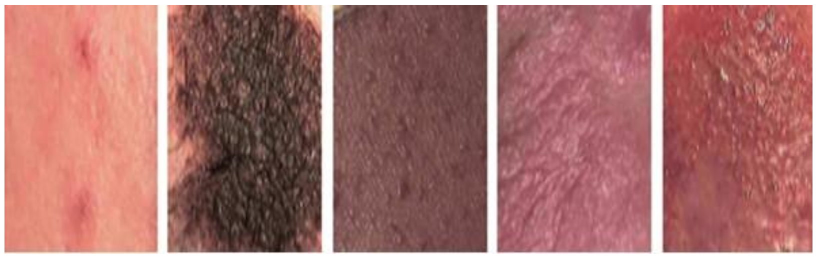


Figure 1.3: Five dermatological disorders [11].

Moving on a little bit further, texture feature has also been used for object recognition. Contour and texture have both shown powerful cues for recognition. By combining contour and texture features [12, 13], the recognition accuracy can be significantly improved. Even in the field of remote sensing, texture classification plays an important role in automatically labeling each region in the image. As Fig. 1.4 shows, textured regions in remotely sensed images such as the urban, industrial and residential area can be classified with some classical texture classification methods [14, 15]. This is very helpful to monitor land use patterns in GIS (Geographic Information System) and predict how they vary with the development of human society.

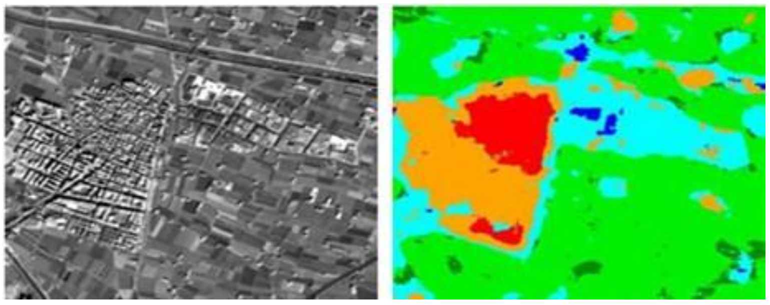


Figure 1.4: Texture classification is applied to a remote sensing image of an urban area [14].

1.2 Overview of Texture Classification Methods

In this section, we briefly review how the research of texture classification evolved in the past three decades. Julesz *et al.*'s early work [119] on the visual perception of texture laid the foundation of texture analysis and synthesis. Since then, plenty of texture classification methods have been proposed. In the early 1980s, the texture classification task was to classify two textures in a binary image, which are synthesized by the repeated placement of basic micro patterns. This classification task intends to verify Julesz *et al*.'s conjecture [119] that two textures cannot be perceptually distinguished if they had the identical second order statistics. Since this conjecture was eventually disproved, the statistical theory of textons [40] was developed, where textons were viewed as fundamental primitives in texture and two textures can be distinguished if they have different texton densities. However, based on this statistical theory of textons, there were still two problems remained. One was how to form a set of textons and the other was how to generalize this theory to the gray scale texture images.

Till the early 1990s, researchers attempted to classify gray scale images of real world textures with the rotation and scale changes. During this period, filter banks [33, 34, 39] were utilized to analyze textures by extracting features at multiple orientations and scales. The mean and variance of filter responses were usually extracted as texture features for classification. Due to the rich representations of filter responses, the filter bank methods can obtain good performance on texture classification. In fact, Julesz *et al.* [121] tried to find a link between textons and filter banks. Since there were some limitations in the theory of textons, in this period filter bank based texture classification methods became very popular.

In the late nineties, real world 3D textures [41] with viewpoint and illumination changes were considered in the texture classification task. Some methods such as texton learning methods [41, 42, 47, 48, 50, 52], invariant filter design methods [122] and fractal methods [20, 21], etc, have been proposed to solve the texture classification problem with large scale and viewpoint changes. Leung and Malik [41] are among the first researchers who attempted to classify 3D textures under varying viewpoints and illuminations. By giving an operational definition of texton based on filter responses and clustering, they found a way to form a set of universal textons for real world textures and bridged the gap between the theory of texton and filter responses. Moreover, based on the theory of Markov random field, Zhu *et al.* [123] gave a mathematical model to define texton and presented a three-level generative image model to learn textons.

In general, texture classification methods mainly consist of two steps: feature extraction and classification. We categorized texture feature extraction methods into three classes: the local descriptor, filter response and texton learning based methods. In local descriptor based texture description methods, the co-occurrence matrix [16] is a classical local descriptor to characterize the local structure in texture. Compared to co-occurrence matrix, local binary pattern (LBP) [17] and its variants [18, 19] are more efficient to describe textures. However, they can’t deal with large scale and viewpoint changes, either. The fractal based descriptor is another simple method to characterize textures and it is robust to certain scale and viewpoint changes. The second class of texture description methods is based on filter responses, e.g., using Gabor filters [23], wavelet filters [24] and steerable pyramid filters [25], etc. Such methods can obtain good performance on textures with some rotation and scale changes, but they do not work well when textures are imaged with large geometrical variations. Different from the above methods, the third class of methods involves a learning stage. Textures are modeled by learned textons and the texton histogram is formed for classification. In these methods, textons are learned from different feature spaces such as filter bank responses [41, 49, 50], original patches [51, 52, 54] and RIFT and SPIN descriptors [47]. For classification, different classifiers such as the Gaussian Bayes classifier [120], nearest neighbor classifier [17, 18, 47, 50, 52, 53] and SVM classifier [48, 54, 55, 56, 90], can be applied to the extracted texture features. In Chapter 2, we will review these texture classification methods with more details.

1.3 Challenges in Texture Classification

Although texture classification has been studied for more than thirty years, there still exist some problems. The most challenging ones are summarized below.

(1) Efficient texture representation. The texture classification and texture analysis can be widely applied to many fields. However, this also makes the problem very hard. Different from other classification problems such as fingerprint, face and indoor scene recognition, where there are clear structures in the image, usually there is no clear large scale structure which can be extracted from texture images. Hence, how to effectively represent texture is a key problem for texture classification.

(2) Robust feature extraction. Texture images often undergo variations in appearance due to the illumination, rotation, scale, and viewpoint changes in image acquisition. Some example images are illustrated in Fig. 1.5. Another factor which

makes the texture classification task very challenging is that it has large inter-class confusion as well as large intra-class variation. Two different classes of texture images captured under different imaging conditions may look very similar in appearance, as is illustrated in Fig. 1.6. Therefore, how to extract robust invariant features to deal with these variations is an important problem in texture classification.

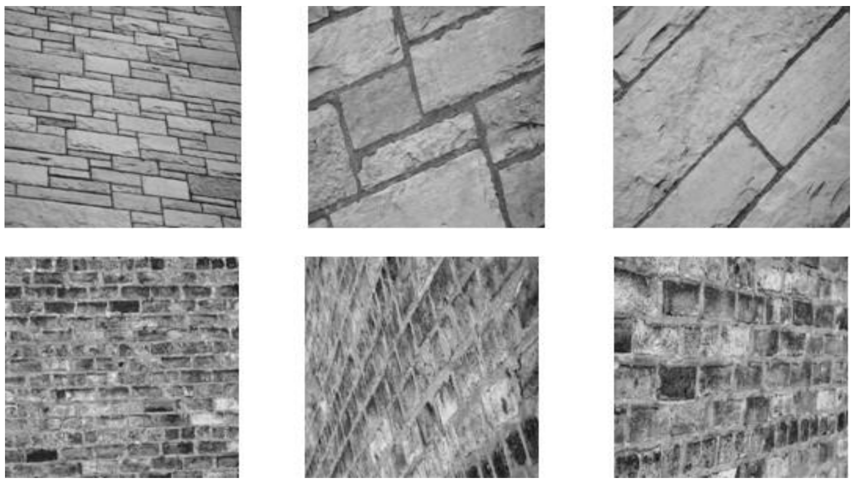


Figure 1.5: Texture images have some scale and viewpoint changes.

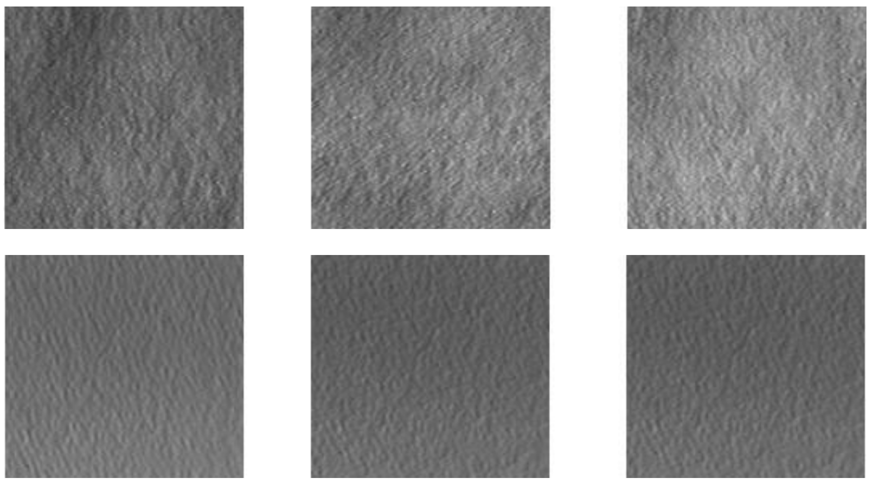


Figure 1.6: Two different kinds of texture samples with similar appearances.

(3) Insufficient training samples. The texture classification performance can drop dramatically with the decrease of the number of training samples even when there are only a small number of classes. In particular, when there are large scale and viewpoint changes, how to improve the classification accuracy with a small number of training images is a very challenging problem.

1.4 Contributions of the Thesis

This thesis makes contributions towards efficient texture representation, effective texture feature extraction and novel applications in hand back skin texture analysis.

(1) A sparse representation (SR) based texton learning method is proposed to learn sparse textons for texture representation. Then the histogram of sparse coding coefficients is extracted as features for texture classification. Traditional texture modeling approaches usually learn textons in the feature space using the *K*-means clustering method. However, the *K*-means clustering may not be able to well characterize the intrinsic feature space of textons, which is often embedded into a low dimensional manifold. Hence, we use the SR based dictionary learning method to learn textons instead of the traditional *K*-means clustering. The SR coefficients of the texture images over the dictionary are used to construct the histograms for classification. Experimental results showed that the proposed method yields good performance.

(2) The SR based texton learning has high computational complexity, while the *K*-means clustering based texton learning has high representation error. In this thesis, we propose a simple and efficient texton learning method based on the regularized least square, which has low computational complexity as well as low

representation error. Meanwhile, we propose a fast texton encoding method to code the texture feature over the learned dictionary of textons, and define two types of texton encoding induced statistical features -- coefficient histogram and residual histogram -- for texture classification. Experimental results demonstrated that the proposed method can obtain probably the best texture classification accuracy so far, especially when the number of training samples is insufficient.

(3) We study the hand back skin texture (HBST) classification problem with applications to personal identification and gender classification. A specially designed imaging system is developed to capture the HBST images, and an HBST image dataset is established, which consists of 1920 images from 80 persons. Then some texton learning based methods are employed to perform the personal identification and gender classification experiments on the established HBST dataset. The results demonstrated that HBST is helpful to aid human identity identification and gender classification. Meanwhile, the established HBST dataset is very challenging and it is a good platform to evaluate the texture analysis methods.

1.5 Outline of the Thesis

The remainder of the thesis is organized as follows. In Chapter 2, we review the previous works on texture classification. In Chapter 3, the proposed texture classification method by SR based texton learning is presented. In Chapter 4, a simple and efficient texton learning method by regularized least square is developed, and texton encoding induced statistical features are proposed for texture classification. In Chapter 5, an HBST image dataset is established and we then apply the texton learning based texture classification methods to recognize the HBST patterns for human identity and gender classification. Finally, in Chapter 6, we present our conclusions and future work.

Chapter 2. Literature Review

There are many computer vision problems related to texture analysis such as texture segmentation [3], texture compression [4], texture synthesis [5], shape from texture [6, 7] and texture classification [16-22]. In this chapter, we focus on works related to texture classification. Although texture classification has been widely used in many fields, a key problem in it is how to describe texture. Moreover, as introduced in Chapter 1, since there are some camera pose changes and illumination variations, the methods which are used to describe texture should be robust to illumination variations and geometric variations such as rotation, scale, viewpoint and deformation changes. During the past decades, researchers have proposed many texture feature extraction methods to solve these problems. These feature extraction methods mainly fall into three categories: local descriptors, filter responses and texton learning. Based on the extracted features, different classifiers can be employed for classification. In this chapter, we review some classical texture classification methods in the three categories.

2.1 Texture Feature based on Local Descriptors

2.1.1 Co-occurrence Matrix

The gray-level co-occurrence matrix developed by Haralick *et al.* [16] is widely used in texture feature extraction. Given an offset (Δ_x, Δ_y) in an $n \times m$ image, the probability of co-occurrence between gray values i and j can be computed for all possible co-occurring grey level pairs in an image window. The co-occurrence matrix stores these probabilities, which are the number of all available grey levels in the image. Then, selected statistics are applied to the co-occurrence matrix by iterating through the entire matrix to calculate the texture features. The co-occurrence matrix can be defined as follows:

$$C(\Delta_x,\Delta_y,i,j)=\sum_{p=1}^{m}\sum_{q=1}^{n}\begin{cases}1, \mathit{if}\ I(p,q)=i\ \mathit{and}\ I(p+\Delta_x,q+\Delta_y)=j\\ 0, \mathit{otherwise}\end{cases} \tag{2-1}$$

Note that the co-occurrence matrix is not invariant to rotation. Hence, in order to obtain the rotation invariance, the co-occurrence matrix is computed by using a set of offsets sweeping 180 degree (i.e., 0, 45, 90 and 135 degrees) at the same distance. Then a set of statistic features based on the co-occurrence matrix are used for texture classification. Although the co-occurrence matrix can characterize the statistics of pixels in the texture image well, the complexity of computing the co-occurrence matrix is very high. Moreover, the co-occurrence matrix is only invariant to rotation. When there are some scale and viewpoint changes, the co-occurrence matrix can’t handle these situations.

2.1.2 Local Binary Pattern (LBP) and Its Main Variants

The local binary pattern (LBP) operator is one of the best local texture descriptors and it has been widely used in various tasks such as texture classification, face recognition, face expression recognition, dynamic texture recognition and human action recognition. In this sub-section, we review the original LBP operator and some major variants for different applications.

The LBP was originally proposed by Ojala *et al.* [17] for texture classification. It is a very simple yet effective local descriptor to describe texture. LBP can characterize the spatial structure in the texture image well and has a good discriminative power. It compares each pixel with its neighborhoods to encode each pixel as a sequence of binary codes:

$$LBP_{P,R} = \sum_{p=0}^{P-1} s(g_p - g_c)2^p,\ s(x) = \begin{cases} 0, & x < 0 \\ 1, & x \geq 0 \end{cases} \qquad (2\text{-}2)$$

where g_c is the gray value of the central pixel in the texture image and g_p represents the gray value of the pixel in the circle neighbor. R is the radius of the neighborhood and P is the number of pixels in the neighborhood. Fig. 2.1 shows the neighborhoods for different R and P. Some points of the neighborhood which are not on the image grid can be estimated by the pixel interpolation method. Fig. 2.2 illustrates the binary encoding process of the LBP operator. If the gray value of the pixel in the neighborhood is smaller than the gray value of the central pixel, it is encoded as “0”. Otherwise, it is encoded as “1”. Then a set of binary codes can be formed to describe the local structure of the texture image.

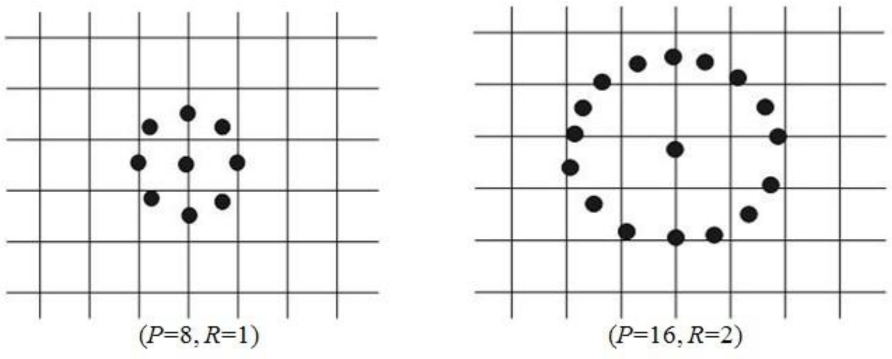


Figure 2.1: Different neighborhoods of the LBP operator.

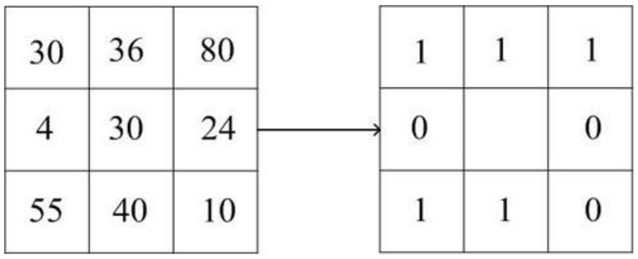


Figure 2.2: Illustration of the encoding process of the LBP operator.

Ojala *et al.* [17] observed that some local binary patterns are the vast majority, sometimes over 90% of all local binary patterns. Hence, a uniform measure U can be defined by calculating the number of spatial transitions in the pattern:

$$U\left(LBP_{P,R}\right)=\left|s\left(g_{p-1}-g_c\right)-s(g_0-g_c)\right|+\sum_{p=0}^{P-1}\left|s\left(g_p-g_c\right)-s\left(g_{p-1}-g_c\right)\right| \quad (2\text{-}3)$$

The uniform patterns are defined as those patterns whose U values are not greater than 2 and the other patterns are called non-uniform ones. Hence, the rotation invariant texture descriptor can be defined as follows:

$$LBP_{P,R}^{riu2}=\begin{cases}\sum_{p=0}^{P-1}s(g_p-g_c),\ if\ U(LBP_{P,R})\leq 2\\ P+1,\ otherwise\end{cases} \quad (2\text{-}4)$$

Ojala *et al.* [17] pointed out that image texture has two complementary characteristics: spatial structure and contrast. Spatial pattern is affected by rotation while contrast is affected by the gray scale. The *LBPriu2* operator can describe the spatial structure of local image texture well, but it ignores the other important characteristic, i.e., contrast. Hence, some variants of the LBP operator are developed by employing more useful information from the local structure of texture image.

Variant 1: ***VAR***. Ojala *et al*. [17] proposed a rotation invariant measure of local variance, namely variance (VAR), to incorporate the contrast of local texture image:

$$VAR_{P,R} = \frac{1}{P}\sum_{p=0}^{P-1} s(g_p - \mu) \quad \mu = \frac{1}{P}\sum_{p=0}^{P-1} g_p \tag{2-5}$$

Experimental results showed that the texture classification performance can be further improved by combining *LBPriu2* with VAR. However, VAR is less effective if the texture image has illumination changes or scale changes.

Variant 2: ***DLBP***. Uniform LBPs may not be the dominant patterns in some texture images due to the irregular edges and shapes. Hence, in [19] the authors generalized the original LBP to the dominant local binary pattern (DLBP). In DLBP, the occurrence frequencies of all rotation invariant patterns are computed and several most frequently occurring patterns are chosen as the dominant patterns. Experimental results demonstrated that the dominant patterns which usually count about 80 percent of the total pattern occurrences in an image can well capture the local texture structure for classification tasks. By extracting the dominant patterns, DLBP is more robust to noise than the original LBP. Furthermore, Liao *et al*. [19] employed additional features based on Gabor filter responses as the supplement to the DLBP features. It is experimentally shown that the fused features can obtain higher classification rate than the DLBP feature.

Variant 3: ***CLBP***. Although *LBPriu2* can describe local structure well and VAR can describe contrast well, the combination of them is not robust to illumination and scale changes. Guo *et al*. [18] proposed a novel descriptor to generalize the original LBP by fully exploiting the local structural information in the texture image, namely completed local binary pattern (CLBP). This descriptor obtains state-of-the-art classification results on many texture datasets. CLBP decomposes the image local difference into two complementary components, the signs (s) and the magnitudes (m):

$$s_p = s(g_p - g_c), m_p = \left|g_p - g_c\right| \tag{2-6}$$

Based on the sign and magnitude information, CLBP-Sign (CLBP_S) and CLBP-Magnitude (CLBP_M) are proposed to encode the local structure in the texture image. Actually, CLBP_S is equivalent to the conventional LBP, and CLBP_M measures the local variance of the magnitude. CLBP_M and CLBP_S are defined in the traditional encoding way of the original LBP as follows:

$$CLBP\_M_{P,R} = \sum\nolimits_{p=0}^{P-1} t(m-c)2^p, t(x,c) = \begin{cases} 0, x < c \\ 1, x \geq c \end{cases} \tag{2-7}$$

Here threshold c is set as the mean value of m_p. The center pixel, which has discriminative information, is encoded by CLBP_C:

$$CLBP\_C_{P,R} = t(g_c - g_I) \tag{2-8}$$

By combining CLBP_C, CLBP_M and CLBP_S in different ways, CLBP can yield much better performance than many LBP based texture classification methods.

2.1.3 Fractal Feature

Although the co-occurrence and LBP features can describe the local structure of texture image well, these features are not suitable for describing texture images with some scale and viewpoint changes. The fractal feature can overcome these problems to some extent. Xu *et al.* [20, 21, 22] proved that multifractal spectrum (MFS) vector is globally invariant under the bi-lipschitz transform.

In [20], the MFS vector, which is the vector of the fractal dimensions of the texture image, is used to describe the texture surface. The concept of the fractal dimension can be extended to the concept of MFS as follows. First a point categorization is defined according to the density function. Then the fractal dimensions are calculated for every point set from this categorization. The defined MFS vector gives a rich description of the inherent texture structure.

The local density function $D(i)$ at position i in the texture image is defined as follows:

$$D(i) = \lim_{r \to 0} \frac{\log \mu(B(i,r))}{\log r} \tag{2-9}$$

where $B(i,r)$ is the disc at center i with radius r and μ is the measurement function on the texture image. In [20], there are three ways to define the measurement function μ: the sum of average intensity values inside $B(i,r)$, the energy of the gradients inside $B(i,r)$ and the sum of the Laplacian inside $B(i,r)$.

For any α, E_α is the set of all image points with local density α:

$$E_\alpha = \{\, i : D(i) = \lim_{r \to 0} \frac{\log \mu(B(i,r))}{\log r} = \alpha \} \tag{2-10}$$

Then, based on the set of all points with local density α, we can compute the fractal dimension to obtain an MFS $f(\alpha)$:

$$f(\alpha) = \{\dim(E_\alpha) : \alpha \in R\} \tag{2-11}$$

From Eq. (2-9), we have:

$$\log \mu(B(i,r)) = D(i)\log r + L(i) \tag{2-12}$$

In [53], the authors observed that the local fractal dimension $D(i)$ and the local fractal length $L(i)$ are very useful to distinguish textures. Hence, the MR8 filter response based fractal features are employed to classify textures. D and L are estimated at each pixel. Particularly, based on empirical results, only five measurements are calculated for D, which is invariant to the bi-lipschiz transform. And the fractal length L is an 8 dimensional vector, which is rotation invariant. From the experimental results, the features D and L can both obtain good performance on current texture datasets.

2.2 Texture Feature based on Filter Responses

Filtering approaches such as the Gabor transform [23], wavelet transform [24], steerable pyramid filter [25] and directional filter bank [27, 28] can provide good multi-resolution analysis for texture classification. Also, the Gabor filter, steerable pyramid filter and directional filter bank can characterize texture in multiple scales and orientations.

The texture analysis with the discrete wavelet transform has two disadvantages: poor directionality and rotation sensitivity (wavelet transform is not rotation invariant). Hence, researchers usually use some preprocessing steps or design special filters to achieve rotation invariant texture features. Pun and Lee [29] used the log-polar transform to convert the rotation and scale variance to the translation variance, and then employed a shift invariant wavelet packet transform to extract rotation and scale invariant features. Khouzani and Zadeh [30] first calculated the randon transform of texture images and then used a translation invariant wavelet transform to obtain rotation invariance. They [31] also proposed a method based on randon transform to estimate the orientation of the texture image and used the wavelet transform for invariant texture analysis. Porter and Canagarajah [32] employed the standard discrete wavelet transform and used the combined opposite energy signatures as features for rotation invariant texture classification. Chen and Kundu [33], and Wu and Wei [34] used the sub-band decomposition and Hidden Markov Model (HMM) to extract rotation and scale invariant features by supervised learning. Do and Vetterli [35] employed HMM in the steerable wavelet domain and a maximum likelihood estimator to obtain rotation invariant texture features. Kim and Upda [36] proposed the rotated wavelet filters for texture classification by rotating the filters to the 45 degree. However, the rotated filters are not invariant to any angles. To overcome the drawbacks of poor directionality of wavelet transform, Kokare *et al.* [37] extracted rotation invariant features by rotated complex wavelet filters for texture classification.

Gabor filter and steerable pyramid filter are alternatives in filtering approaches. Haley and Manjunath [38] employed a complete space-frequency Gabor wavelet model for rotation invariant texture classification. The steerable pyramid filters proposed by Simoncelli [25, 26] have the ability to decompose a signal into multi-resolution directional sub-bands. Greenspan [39] used this kind of steerable orientation filters to extract texture features. Gabor filter and steerable pyramid filter, however, are redundant in both scale and directional decomposition, which results in high-computational complexity for feature extraction.

The main drawback of filter response based feature extraction method is that the texture classification performance is not good in the case of large scale and viewpoint changes. Although some filters are specially designed for rotation and scale invariance, the filter responses of texture images are not robust enough to deal with large scale and viewpoint changes.

2.3 Texture Feature based on Texton Learning

2.3.1 Leung and Malik’s Algorithm

The concept of texton was proposed by Julesz [40] nearly 30 years ago, which is the putative unit of pre-attentive human texture perception. Leung and Malik [41] built a small and finite vocabulary of micro-structures for 3D texture classification, which is called 3D textons. The goal is to use some training samples to learn a dictionary which can describe all classes of textures. For each class of texture images, 20 training images with different lighting and viewpoint angles are used to build the dictionary of textons. The training images are registered using the sum-of-square-differences (SSD) algorithm. Then these registered training images are filtered by a 48 dimensional filter bank to generate filter response features. During learning, these filter responses are combined together to span a 960 ($20\times48=960$) dimensional space. Then the filter response vectors (960-dimension) are clustered using the *K*-means clustering to determine some cluster centers, which are called 3D texons. These learned 3D textons encode the appearances of texture images for

3D texture classification. For 3D textons, all filter response vectors are labeled with the texton which is closest to them in the filter response space. The texton histogram, which is the distribution of texton frequencies, can be used as the texture model.

In the stage of classification, the testing images will go through the same procedures to generate texton histograms. For each testing texture image, filter response features can be generated using the above mentioned 48 dimensional filter bank. The texton histogram can be formed and the distance between the histograms can be computed using the Chi-square distance. Finally, classification is performed on the texture dataset using the nearest neighbor classifier.

The authors also developed a Markov Chain Monte Carlo (MCMC) algorithm to classify a single image under known imaging conditions. Each pixel in the texture image is labeled by some possible textons in the dictionary. Then the MCMC algorithm is used to find the most possible label with the given possibility. However, the classification accuracy of the algorithm is not as good as that by multiple texture training images.

2.3.2 Cula and Dana's Algorithm

Cula and Dana [42, 43, 44] extended Leung and Malik's algorithm and illustrated that 2D textons can also be utilized for un-calibrated and single texture image classification and can obtain comparable performance.

The overall methodology is the same as Leung and Malik's except that the occurrences of 3D textons are replaced by 2D textons. For example, during the process of texton learning, training texture images are filtered by a set of filter banks across different scales and clustered immediately without being concatenated. The texton histogram can be formed by labeling the filter responses of each pixel in the texture image. Moreover, multiple models can be used for each texture class. As in Leung and Malik's algorithm, the nearest neighbor classifier is employed to classify textures by matching the texton histograms.

Since the texton histogram space is high dimensional, a projection of this high dimensional histogram to a low dimensional one is desired, which is expected to preserve the statistical properties of the high dimensional histogram. To accomplish dimension reduction of the high dimensional texton histogram, the authors employed principal component analysis (PCA). However, this PCA based method to reduce the number of texton models ignores the inter class variation between textures. Hence, the classification accuracy by the reduced models is not good. Experiments were conducted on 156 images per class in the CUReT dataset [89]. If 56 models are chosen for training, a classification accuracy of 96% can be achieved. If 8 models are chosen per class, however, only a classification accuracy of 71% can be achieved.

2.3.3 Lazebnik *et al.*'s Algorithm

As described in Chapter 1, texture images often undergo some large scale and viewpoint changes. Similar to the SIFT descriptor [46], Lazebnik *et al.* [47] used some affine invariant detector [45] to detect some affine invariant regions and then formed the descriptor on these detected regions. These descriptors are then used to learn textons to classify texture images. The main difference between Lazebnik *et* *al.*'s algorithm and other texton learning based algorithms is that a set of local textons are learned per image during the training and testing process.

The main steps of the algorithm are as follows. The interest points are detected in the texture image using the Harris-affine, Hessian-affine and Laplacian of Gaussian detectors. For each interest point, a characteristic scale is determined by finding the local maximum or minimum in the neighbors across different scales. And the surrounding region of the detected interest point is normalized by the characteristic scale and main orientation to form the affine invariant region. Spin and RIFT images are then used to form the texture descriptor on the detected regions. These descriptors are then clustered by the *K*-means clustering algorithm to learn textons. Then the texture image is labeled using the learned textons and the texton histogram is formed to classify textures.

During classification, the Earth Mover's Distance (EMD) [59] is used to compute the distance between the histograms. Computing the EMD is performed on the Harris-affine interest points and the LOG interest points separately. The nearest neighbor classifier is used for classification.

2.3.4 Varma and Zisserman's Algorithm

In Leung and Malik's work, 3D textons are used to classify texture images. In the learning stage, training samples of each class should be registered. Leung and Malik also developed an MCMC algorithm for classifying a single texture image under known imaging conditions. However, the classification accuracy is only 87%, which is not very satisfying. Different from Leung and Malik's method, in Varma and Zisserman's method [49, 50], the texton clustering is in an extremely low dimensional space which is invariant to rotation by using the MR8 filter bank [49]. Consequently, the texton histogram computed in the MR8 space can obtain better classification accuracy than the method in [41] using 3D texton.

In Varma and Zisserman's method, multiple and unregistered images of the same texture are filtered by MR8 filter bank. The MR8 filter bank is a rotationally invariant, nonlinear filter bank with 38 filters but only 8 filter responses. It contains bar and edge filters, each at 6 orientations and 3 scales, as well as a Gaussian and a Laplacian of Gaussian filter. The filter responses of training samples of each class are aggregated and clustered into some textons using the *K*-means method. Then textons from different texture classes are combined to form the dictionary of textons. To characterize various texture classes, models are generated by labeling each of their filter responses with the texton that is closest to it in filter response space. Thus each of the training images is quantized into a texton map. Finally, the texton histogram of the texton map is used to form a model corresponding to the particular training texture image. In the classification stage, the same procedures are applied to build the texton histogram of the testing texture image. The nearest neighbor classifier with the Chi-square distance is used for classification.

Furthermore, Varma *et al*. [51, 52] observed that filter responses can be replaced with the source image patches. The main reason is that convolution to generate filter responses can be rewritten as an inner product between image patch vectors and the filter bank. Hence, the filter response is essentially a lower dimensional projection of an image patch into a linear space spanned by the set of filters. Instead of using a filter bank to generate filter responses at each pixel, the intensities of a square neighborhood around that pixel are taken as a vector. Then textons are clustered in image patch space and the texton histogram can be formed for classification. Experimental results showed that the image patch vector feature can lead to comparable or even better results than the MR8 feature in terms of classification accuracy.

2.4 Classification

Once texture features are extracted, different classifiers can be employed for classification. In classical texture classification methods, the commonly used classifiers are the nearest neighbor classifier, SVM classifier, and the Gaussian Bayesian classifier.

In [17, 18, 47, 50, 52, 53], the nearest neighbor classifier is used for final classification. A testing texture image can be classified into the corresponding class based on the closest distance between the testing sample and the training samples. Usually, there are two distance functions: the χ^2 distance [17, 18, 50, 52] and the Earth Mover's distance (EMD) [47, 48]. For two histograms $\boldsymbol{H}_1$ and $\boldsymbol{H}_2$, the χ^2 distance can be defined:

$$\chi^2(\boldsymbol{H}_1, \boldsymbol{H}_2) = \frac{1}{2}\sum_i \frac{(\boldsymbol{H}_1(i) - \boldsymbol{H}_2(i))^2}{\boldsymbol{H}_1(i) + \boldsymbol{H}_2(i)} \tag{2-13}$$

The EMD is suitable to measure the similarity between image signatures. In texton learning based texture classification methods, the image signature $\{(\boldsymbol{p}_1, \mu_1), (\boldsymbol{p}_2, \mu_2), ..., (\boldsymbol{p}_m, \mu_m)\}$ (where m is the number of clusters, $\boldsymbol{p}_i$ is the center of cluster i, u_i is the relative size of cluster i) can be formed by clustering the set of descriptors in the texture image. The EMD between two signatures

$\boldsymbol{S}_1 = \{(\boldsymbol{p}_1, \mu_1), (\boldsymbol{p}_2, \mu_2), ..., (\boldsymbol{p}_m, \mu_m)\}$ and $\boldsymbol{S}_2 = \{(\boldsymbol{q}_1, \nu_1), (\boldsymbol{q}_2, \nu_2), ..., (\boldsymbol{q}_n, \nu_n)\}$ is defined as follows:

$$EMD(\boldsymbol{S}_1, \boldsymbol{S}_2) = \frac{\sum_{i=1}^{m}\sum_{j=1}^{n} f_{ij} d(\boldsymbol{p}_i, \boldsymbol{q}_j)}{\sum_{i=1}^{m}\sum_{j=1}^{n} f_{ij}} \tag{2-14}$$

where f_{ij} is a flow value that can be solved by linear programming, and $d(\boldsymbol{p}_i, \boldsymbol{q}_j)$ is the ground distance between cluster centers $\boldsymbol{p}_i$ and $\boldsymbol{q}_j$.

SVM [48, 54, 55, 56, 90] is another widely used classifier in texture classification. In a two-class classification problem, the decision function for a testing sample $\boldsymbol{x}$ is defined as follows:

$$g(\boldsymbol{x}) = \sum_i \boldsymbol{\alpha}_i \boldsymbol{y}_i K(\boldsymbol{x}_i, \boldsymbol{x}) - \boldsymbol{b} \tag{2-15}$$

where $K(\boldsymbol{x}_i, \boldsymbol{x})$ is a kernel function for the training sample $\boldsymbol{x}_i$ and the testing sample $\boldsymbol{x}$, $\boldsymbol{y}_i$ is the class label of the training sample (+1 or -1), $\boldsymbol{\alpha}_i$ is the learned weight of the training sample, and $\boldsymbol{b}$ is a threshold. These training samples with the weight $\boldsymbol{\alpha}_i > 0$ are called “support vectors”. The binary SVM can be extended to the multi-classification problem with different methods such as the one-against-one technique. In [48, 55, 90], the Gaussian kernel is chosen for the SVM classifier. Hayman *et al.* [90] showed that the classification accuracy can be improved with the SVM classifier than the nearest neighbor classifier. Moreover, the average number of support vectors used is 10%-20% lower than the number of models required by the nearest neighbor classifier.

In addition, the Gaussian Bayesian classifier is also used in [120] for texture classification instead of the nearest neighbor classifier. By computing a parametric estimate of the variation of marginal histograms with the multivariate Gaussian distributions, two Gaussian classifiers, which can model the variation of the marginal histograms, are used independently and jointly for classification.

2.5 Summary

In this chapter, we reviewed classical and representative texture feature extraction methods for texture classification. We categorized these methods into three classes. In the local descriptor based texture description methods, LBP is a popular method but it cannot deal with large scale and viewpoint changes. The fractal method is also a simple method to characterize texture and is robust to certain scale and viewpoint changes. However, it is not suitable for low resolution texture images. The second class of texture description methods is based on the filter responses, which can obtain good performance on textures with some rotations. However, the texture features based on filter responses cannot work well when texture images have large geometrical changes. Different from the above methods, the third class of methods involves a learning stage. Textures are modeled by learned textons and the texton histogram is formed for classification.

Chapter 3. Texture Classification via Sparse Texton Learning

In this chapter, we propose a new texture classification method via sparse texton learning. In Section 3.1, we briefly review existing representative texton learning methods for texture classification. An overview of sparse representation of signals and algorithms to solve the l_1-norm optimization problem is described in Section 3.2. In Section 3.3, we propose a sparse texton learning method to represent texture and the representation coefficient histogram is constructed for texture classification. In Section 3.4, the proposed method is validated on the CUReT and KTH_TIPS texture datasets. Finally, the conclusion is made in Section 3.5.

3.1 Introduction

In order to perform texture classification with large viewpoint and scale changes, Leung and Malik [41] first investigated the possibility to build a set of universal textons for real world textures by learning textons from the filter response spaces. They were the first researchers who attempted to classify 3D textures under varying viewpoint and illumination changes with the texton learning method. Since then, many texton learning based methods have been proposed. Leung and Malik [41] built a small and finite vocabulary of micro-structures, i.e., textons, for 3D texture classification. These 3D textons are cluster centers of filter responses by applying the *K*-means clustering method over a stack of images with representative viewpoints and illuminations. Cula and Dana [44] extended Leung and Malik's algorithm and showed that 2D textons can be utilized for un-calibrated and single texture image classification with good performance.

Lazebnik *et al*. [47] employed the Harris-affine detector and Laplacian detector to detect the invariant regions in texture images. A combination of SPIN and RIFT descriptors was used to represent these detected regions, and the descriptors are clustered by the *K*-means clustering. The texture image is labeled using the learned textons (i.e., cluster centers) and the texton histogram is computed to classify textures. During classification, the Earth Mover's distance (EMD) is used to compute the distance between histograms. Based on Lazebnik *et al*.'s method, Zhang *et al.* [48] combined three kinds of descriptors, SIFT, SPIN and RIFT, to learn textons and a kernel SVM classifier was used for classification.

Varma and Zisserman [50] modeled texture images as distributions over a set of textons, which are learned from the responses of MR8 filter bank. With the rotation invariant MR8 filter bank, texton clustering can be conducted in an 8 dimensional space. Furthermore, in [52] good performance is achieved by textons learned from patches in the original image instead of MR8 filter responses. In order to deal with large scale and viewpoint changes of texture image, Varma and Garg [53] extracted the local fractal dimension and length from MR8 filter responses to learn textons for classification.

Based on the compressive sensing theory [57, 58], recently Liu *et al*. [54, 55, 56] proposed to use random projection and texton learning for texture classification. First, sorted patch vectors from the original texture image are projected to a low dimensional space with random projection, and then textons are learned with the *K*-

means clustering method in the compressed domain rather than the original patch domain. The process of texture description with learned textons is the same as that in Varma and Zisserman's method [52].

In the above texton based methods, the textons are usually learned by the *K*-means clustering algorithm. However, the *K*-means clustering algorithm is based on the l_2-norm Euclidean distance so that the elements of a cluster will have a ball-like distribution. The learned *K* ball-like clusters, nonetheless, may not be able to characterize reasonably well the intrinsic feature space of the texture images, which is often embedded into a lower dimensional manifold.

Recently, the theory and algorithms of sparse coding or sparse representation (SR) [61, 62, 63, 64] have been successfully used in image processing and pattern recognition [65-72]. The principle of SR reveals that a given natural signal can be often sparsely represented as the linear combination of an over-complete dictionary. Inspired by the great success of SR, in this Chapter we propose a sparse texton learning method for texture classification. A texton training dataset is first constructed from descriptors in the training images, and then an over-complete dictionary of textons is computed under the SR framework. A histogram feature of SR coefficients can be extracted for texture classification by coding the texture image with the texton dictionary. It will be seen that the proposed method can achieve better texture classification performance than the state-of-the-art texton based texture classification methods using textons learned by the *K*-means clustering.

3.2 Sparse Representation of Signals

In recent years there has been a growing interest in the study of SR of signals. The success of SR largely owes to the fact that natural signals are intrinsically sparse in some domain. Wavelet transform can be viewed as a special case of SR, which is a good tool of time-frequency analysis. However, wavelet transform is limited in characterizing the various local structures in natural images. In order to overcome the shortcomings of wavelet transform, more advanced multi-scale transforms such as Ridgelet [75], Curvelet [76, 77], Contourlet [27] and Bandlet [74] are developed. Although these transforms can offer multi-scale and multi-direction representations for natural images, these representations are not adaptive to the image contents. They can only handle some specific classes of images optimally such as piece-wise smooth images, while natural images often have sharp edge structures. If a redundant dictionary of bases can be learned from example images, a good representation of the input signal can be expected by using this redundant dictionary. SR and the associated dictionary learning techniques can be employed to this end.

For a given signal $\boldsymbol{x} \in \boldsymbol{R}^m$, we say that $\boldsymbol{x}$ has a sparse approximation over a dictionary $\boldsymbol{D} = [\boldsymbol{d}_1, \boldsymbol{d}_2, ..., \boldsymbol{d}_l] \in \boldsymbol{R}^{m\times l}$, if we can find a linear combination of only “a few” atoms from $\boldsymbol{D}$ that is “close” enough to the signal $\boldsymbol{x}$. Under this assumption, the sparsest representation of $\boldsymbol{x}$ over $\boldsymbol{D}$ is the solution of

$$\min_\alpha \|\boldsymbol{\alpha}\|_{l_p} \; s.t. \|\boldsymbol{x} - \boldsymbol{D\alpha}\|_2^2 \le \varepsilon \tag{3-1}$$

where $\|\boldsymbol{\alpha}\|_{l_p}$ is an l_p-norm sparse regularization term imposed on $\boldsymbol{\alpha}$, and ε is a small positive constant controlling the representation accuracy.

When $p = 0$, finding the representation of $\boldsymbol{x}$ can be done by solving the following sparse approximation problem:

$$\min_{\alpha} \|\boldsymbol{\alpha}\|_0 \ s.t. \ \|\boldsymbol{x} - \boldsymbol{D\alpha}\|_2^2 \leq \varepsilon \tag{3-2}$$

where $\|\boldsymbol{\alpha}\|_0$ counts the number of non-zeros in the coding vector $\boldsymbol{\alpha}$. The process of solving the above optimization problem is commonly referred to as "sparse coding". However, the l_0-norm sparse coding is a non-convex and NP-hard problem, and approximate solutions are often found by algorithms such as matching pursuit (MP) [78] and orthogonal matching pursuit (OMP) [79].

In practice, the l_1-norm is widely used to replace the l_0-norm in (3-2) for sparse coding, leading to the following optimization problem:

$$\min_{\alpha} \|\boldsymbol{\alpha}\|_1 \ s.t. \ \|\boldsymbol{x} - \boldsymbol{D\alpha}\|_2^2 \leq \varepsilon \tag{3-3}$$

Some greedy algorithms such as basis pursuit [61] and least angle regression (LARS) [80] have been proposed to solve this l_1-minimization problem. Although these greedy algorithms can work well when $\boldsymbol{\alpha}$ is very sparse, when the number of non-zero entries in $\boldsymbol{\alpha}$ increases, the solution of the above optimization problem will become more and more inaccurate. In order for a more efficient solution, some novel algorithms have been proposed, including gradient projection (GP) [81, 82], homotopy [83, 84], iterative shrinkage-thresholding (IST) [85], proximal gradient (PG) [86], and augmented lagrange multiplier (ALM) [87].

The selection of dictionary $\boldsymbol{D}$ plays an important role in sparse coding. The wavelet, curvelet and contourlet bases can be used as the dictionary to represent signals. However, these analytically designed universal dictionaries are too generic to be effective enough for a specific task, such as texture representation. In many applications of computer vision and pattern classification, we may have training samples, and a more effective dictionary $\boldsymbol{D}$ can be learned from a training dataset, denoted by $\boldsymbol{X} = [\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_n] \in R^{m \times n}$. It is expected that each training sample (i.e., each column of $\boldsymbol{X}$) can be sparsely and faithfully represented over the dictionary $\boldsymbol{D}$, i.e., $\boldsymbol{x}_i \approx \boldsymbol{D}\boldsymbol{\alpha}_i$ and only a few elements in $\boldsymbol{\alpha}_i$ are significant. The dictionary $\boldsymbol{D}$, as well as the coding vector $\boldsymbol{\alpha}_i$, can be solved by optimizing the following objective function

$$\min_{\boldsymbol{D},\boldsymbol{\Lambda}} \left\|\boldsymbol{\Lambda}\right\|_1 \; s.t. \left\|\boldsymbol{X} - \boldsymbol{D}\boldsymbol{\Lambda}\right\|_2^2 \le \varepsilon \tag{3-4}$$

where $\boldsymbol{\Lambda} = [\boldsymbol{\alpha}_1, \boldsymbol{\alpha}_2, ..., \boldsymbol{\alpha}_n]$. Problem (3-4) can be solved by alternatively optimizing $\boldsymbol{D}$ and $\boldsymbol{\Lambda}$. Fixing $\boldsymbol{D}$, $\boldsymbol{\Lambda}$ can be solved by the standard l_1-norm optimization methods. Once $\boldsymbol{\Lambda}$ is obtained, the dictionary $\boldsymbol{D}$ can be updated. Various dictionary learning methods have been proposed such as the *K*-SVD [64] and dual Lagrange methods [112].

3.3 Texture Classification via Sparse Texton Learning

In this section, we propose to learn the dictionary of textons under the SR framework, and then use the learned textons to extract histogram features for texture classification. In our work, the textons are learned in the filtering response feature space or the patch based feature space (the MR8 filter bank response and original image patch are used as the texture feature). Therefore, in the following sub-sections, we first briefly introduce the pre-processing for training dataset construction and extracted texture features, and then present the details of sparse texton learning and texture classification.

3.3.1 Texture Feature Extraction

Before learning textons, all texture images are converted to grey level images and are normalized to have zero mean and unit standard deviation. The normalization offers certain amount of invariance to the illumination changes. We can use the MR8 filter bank or the original image patch vector to extract texture feature. These texture features are used to learn textons by the SR techniques introduced in Section 3.2.

The MR8 filter bank [49, 50] is a nonlinear filter bank with 38 filters but only 8 filter responses. It contains 36 bar and edge filters, which are at 6 orientations and across 3 scales, as well as a Gaussian filter and a Laplacian of Gaussian filter at the single scale. In order to obtain rotation invariance, for the edge and bar filters, the maximum filtering response at 6 orientations is selected for each scale. Moreover, using only the maximum orientation response can reduce the number of responses from 38 to 8. Fig. 3.1 illustrates the MR8 filter banks. The motivation for using the MR8 filter bank is to extract rotation invariant texture features. The MR8 filter bank responses are rotation invariant while preserving the distinctive features of the texton images. After computing the MR8 filter responses, the filter response $\boldsymbol{x}_i$ at pixel i is normalized using the Weber's law [49]: $\boldsymbol{x}_i \log(1+L/0.03)/L$, where $L=\|\boldsymbol{x}_i\|_2$ is the magnitude of the filter response vector $\boldsymbol{x}_i$.

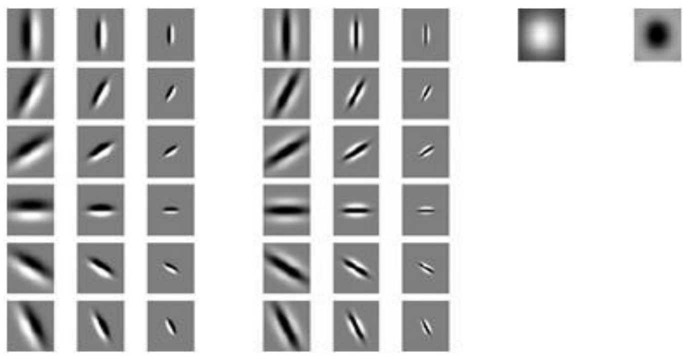


Figure 3.1: The MR8 filter banks.

The second type of feature we can use to learn the textons is the image patch vector [51, 52]. A square neighborhood around each pixel in the image is taken and a vector is formed along the row, as illustrated in Fig. 3.2. In addition, patch vectors are contrast normalized using the Weber's law. Hence, for each class of texture images, we can construct a training dataset $\boldsymbol{X} = [\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_n]$, where $\boldsymbol{x}_i$, $i = 1, 2, ..., n$, is the MR8 filter response vector or patch vector at a position in a training sample image of this class.

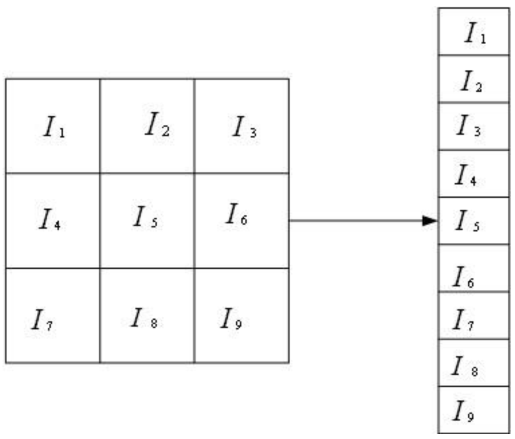


Figure 3.2: The neighborhood is reordered to form a patch vector in row.

3.3.2 Sparse Texton Learning

The dictionary of textons, denoted by $\boldsymbol{D}=[\boldsymbol{d}_1,\boldsymbol{d}_2,...,\boldsymbol{d}_l]$, can be learned from the constructed training dataset $\boldsymbol{X}$, where $\boldsymbol{d}_j$, $j=1,2,...,l$, is one of the l textons. In [49, 50], the classical *K*-means clustering method was employed to determine the l textons by solving the following problem:

$$\min_{\boldsymbol{d}_j}\sum_{j=1}^{l}\sum_{\boldsymbol{x}_i\in\boldsymbol{\Omega}_j}\left\|\boldsymbol{x}_i-\boldsymbol{d}_j\right\|_2^2 \tag{3-5}$$

Obviously, the *K*-means clustering will partition the dataset $\boldsymbol{X}$ into l groups $\boldsymbol{\Omega}_1,\boldsymbol{\Omega}_2,...,\boldsymbol{\Omega}_l$, and the texton $\boldsymbol{d}_j$ is defined as the mean vector of the vectors within $\boldsymbol{\Omega}_j$. However, as we explained in Section 3.1, by using *K*-means clustering, the elements which belong to the same cluster will distribute within a ball because the l_2-norm Euclidean distance is used in the clustering process. These ball-like clusters will cover the whole feature space. Nonetheless, such a dense coverage may not be able to effectively characterize the intrinsic feature space of texture images, which is often embedded into a lower dimensional manifold.

Let $\boldsymbol{\Lambda}=[\boldsymbol{\alpha}_1,\boldsymbol{\alpha}_2,...,\boldsymbol{\alpha}_n]$, the SR objective function in Eq. (3-4) is adopted to optimize $\boldsymbol{D}$ and $\boldsymbol{\Lambda}$, and here we re-write it as follows by setting $p=1$:

$$\min_{\boldsymbol{D},\boldsymbol{\Lambda}}\left\|\boldsymbol{\Lambda}\right\|_1 \ s.t.\left\|\boldsymbol{X}-\boldsymbol{D}\boldsymbol{\Lambda}\right\|_F^2\le\varepsilon \tag{3-6}$$

In practice, it is more convenient to convert Eq. (3-6) into an unconstrained optimization problem by using a form of l_1-penalized least-squares:

$$\min_{\boldsymbol{D},\boldsymbol{\Lambda}}\left\|\boldsymbol{X}-\boldsymbol{D}\boldsymbol{\Lambda}\right\|_F^2+\lambda\left\|\boldsymbol{\Lambda}\right\|_1 \tag{3-7}$$

Eqs. (3-6) and (3-7) are equivalent with an appropriate parameter λ, which is used to balance the l_1-norm and l_2-norm terms in Eq. (3-7). We adopted the truncated Newton interior point method (TNIPM) [82] and the dual Lagrange method [112] to optimize this objective function alternatively. Fixing $\boldsymbol{D}$, $\boldsymbol{\Lambda}$ can be solved by the TNIPM method. In the TNIPM method, the logarithmic barrier function $\Phi(\boldsymbol{\alpha}_i, \boldsymbol{\mu}_i)$ for the constraints $-\mu_{ij} \le \alpha_{ij} \le \mu_{ij}\,(j = 1, 2, ..., n)$ is constructed by

$$\Phi(\boldsymbol{\alpha}_i, \boldsymbol{\mu}_i) = -\sum_j \log(\mu_{ij} + \alpha_{ij}) - \sum_j \log(\mu_{ij} - \alpha_{ij}) \tag{3-8}$$

where $\boldsymbol{\alpha}_i = (\alpha_{i1}, \alpha_{i2}, ..., \alpha_{il})^T$ and $\boldsymbol{\mu}_i = (\mu_{i1}, \mu_{i2}, ..., \mu_{il})^T$. Over the domain of $(\boldsymbol{\alpha}_i, \boldsymbol{\mu}_i)$, the central path consists of the unique optimal solution $(\boldsymbol{\alpha}_i^*(t), \boldsymbol{\mu}_i^*(t))$ of the convex function

$$F_t(\boldsymbol{\alpha}_i, \boldsymbol{\mu}_i) = t(\|\boldsymbol{x}_i - \boldsymbol{D}\boldsymbol{\alpha}_i\|_2^2 + \lambda \sum\nolimits_{j=1}^{n} \mu_{ij}) + \Phi(\boldsymbol{\alpha}_i, \boldsymbol{\mu}_i) \tag{3-9}$$

where parameter $t \in [0, \infty)$. Using the primal barrier method, the optimal search direction with Newton's method is computed by

$$\nabla^2 F_t(\boldsymbol{\alpha}_i, \boldsymbol{\mu}_i) \cdot \begin{pmatrix} \Delta\boldsymbol{\alpha}_i \\ \Delta\boldsymbol{\mu}_i \end{pmatrix} = -\nabla F_t(\boldsymbol{\alpha}_i, \boldsymbol{\mu}_i) \tag{3-10}$$

Fixing $\boldsymbol{\Lambda}$, $\boldsymbol{D}$ can be optimized using the simple dual Lagrange method [112]:

$$\boldsymbol{D}^T = (\boldsymbol{\Lambda}\boldsymbol{\Lambda}^T + \boldsymbol{P})^{-1}(\boldsymbol{X}\boldsymbol{\Lambda}^T)^T \tag{3-11}$$

where $\boldsymbol{P}$ is a diagonal matrix of Lagrange multipliers, which can be obtained by maximizing the dual Lagrange function.

In some sense, the *K*-means clustering method can be viewed as a special case of the SR based clustering in Eq. (3-2). If we let $\boldsymbol{\alpha}_i$ has only one non-zero element and let this non-zero element be 1, then Eq. (3-2) will be basically the same as Eq. (3-5). In this case, we use only one texton to represent the feature vector $\boldsymbol{x}_i$ and assign the label of $\boldsymbol{x}_i$ to that texton. In contrast, by using SR, $\boldsymbol{x}_i$ or any input vector $\boldsymbol{y}$ will be coded as a linear combination of more than one texton. Therefore, SR can achieve a much lower reconstruction error due to the less restrictive constraint. In addition, for an input vector $\boldsymbol{y}$ which may lie in the boundary of two or more clusters, the *K*-means clustering will randomly assign it to one of the classes. Such a representation may not be efficient enough in practice. In the experiments in Section 3.4, we will see that by using the SR technique to learn the textons and using the associated feature description method in Section 3.3.3, the texture classification accuracy can be improved.

3.3.3 Feature Description and Texture Classification

Denote by $\boldsymbol{D}_k$ the texton dictionary for the k^{th} texture class, the dictionary for all the c classes of texture images can be formed by amalgamating the c dictionaries: $\boldsymbol{D} = [\boldsymbol{D}_1, \boldsymbol{D}_2, ..., \boldsymbol{D}_c]$. With this dictionary $\boldsymbol{D}$, each training texture image can generate a model by mapping it to the texton dictionary. In Varma and Zisserman's method [50, 52], for each position of a training image, it is labeled with the elements in the texton dictionary $\boldsymbol{D}$ that is closest to the feature vector at this position. Therefore, a histogram can be formed by normalizing the frequencies of texton labels of this image.

Different from the method in [50, 52], we can construct a histogram of the SR coefficients of a training image as the texture model. Denote by $\boldsymbol{x}_i$ the image feature vector at position i of a training image, we can represent $\boldsymbol{x}_i$ over $\boldsymbol{D}$ by SR to get the representation coefficient vector. However, this can be very computationally expensive because $\boldsymbol{D}$ can be very big. Specifically, we use the closest t textons ($t << z$) to $\boldsymbol{x}_i$ in $\boldsymbol{D}$ to form the sub-dictionary for $\boldsymbol{x}_i$. Denote by $\boldsymbol{d}_1^i, \boldsymbol{d}_2^i, ..., \boldsymbol{d}_t^i$ the t closest textons to $\boldsymbol{x}_i$, the sub-dictionary for $\boldsymbol{x}_i$ is then $\boldsymbol{D}_i = [\boldsymbol{d}_1^i, \boldsymbol{d}_2^i, ..., \boldsymbol{d}_t^i]$. The representation vector of $\boldsymbol{x}_i$ over $\boldsymbol{D}_i$, denoted by $\boldsymbol{\alpha}_i = [\boldsymbol{\alpha}_1^i, \boldsymbol{\alpha}_2^i, ..., \boldsymbol{\alpha}_t^i]$, can then be computed by solving the following l_1-norm minimization problem:

$$\min_{\boldsymbol{\alpha}_i} \left\| \boldsymbol{x}_i - \boldsymbol{D}_i \boldsymbol{\alpha}_i \right\|_2^2 + \lambda \left\| \boldsymbol{\alpha}_i \right\|_1 \tag{3-12}$$

The l_1-least square method in [82] can be used to solve Eq. (3-12).

Since the textons $\boldsymbol{d}_1^i, \boldsymbol{d}_2^i, ..., \boldsymbol{d}_t^i$ in $\boldsymbol{D}_i$ have a one-to-one correspondence to the textons in $\boldsymbol{D}$, by using $\boldsymbol{\alpha}_i$ we can easily construct another representation vector $\boldsymbol{h}_i$ of $\boldsymbol{x}_i$ over $\boldsymbol{D}$ such that

$$\boldsymbol{D}_i \cdot \boldsymbol{\alpha}_i = \boldsymbol{D} \cdot \boldsymbol{h}_i \tag{3-13}$$

Obviously, most of the entries in $\boldsymbol{h}_i$ will be 0, and only the entries corresponding to the same textons as those in $\boldsymbol{D}_i$ will have non-zeros values, and these values are the same as those in $\boldsymbol{\alpha}_i$.

Finally, at each position i of a training texture image, we have a representation vector $\boldsymbol{h}_i$. Then we can form a vector, denoted by $\boldsymbol{h}_f$, for this texture image by summing all the vectors of $\boldsymbol{h}_i$:

$$\boldsymbol{h}_f = \sum \boldsymbol{h}_i \tag{3-14}$$

The histogram $\boldsymbol{h}_f$ can serve as the texture model.

We denote by $\boldsymbol{m}_i$, $i = 1, 2, ..., n$, the model histograms in the dataset. Similarly, for an input testing image $\boldsymbol{y}$, we can construct the sparse texton histogram for it, denoted by $\boldsymbol{h}_y$. The similarity between $\boldsymbol{m}_i$ and $\boldsymbol{h}_y$ is computed as:

$$\chi^2(\boldsymbol{m}_i, \boldsymbol{h}_y) = \frac{1}{2}\sum_j \frac{(\boldsymbol{m}_i(j) - \boldsymbol{h}_y(j))^2}{\boldsymbol{m}_i(j) + \boldsymbol{h}_y(j)} \tag{3-15}$$

The texture image $\boldsymbol{y}$ is classified to the corresponding texture class by the nearest neighbor classifier.

3.4 Experimental Results

In this section, we evaluate the proposed texture classification method on the CUReT [89] and KTH_TIPS [90] datasets. The CUReT texture dataset contains 61 classes, each consisting of 205 images. Here we choose 92 images per class in which a large region of texture is visible. Fig. 3.3 shows some example texture images from the CUReT dataset. There are a number of factors that make the CUReT texture dataset challenging. It has both large inter-class confusion and intra-class variation. The images are obtained under unknown viewpoint and illumination conditions, and some classes look similar in appearance. Figs. 3.4 (a) and (b) show two different classes of textures who look very similar.

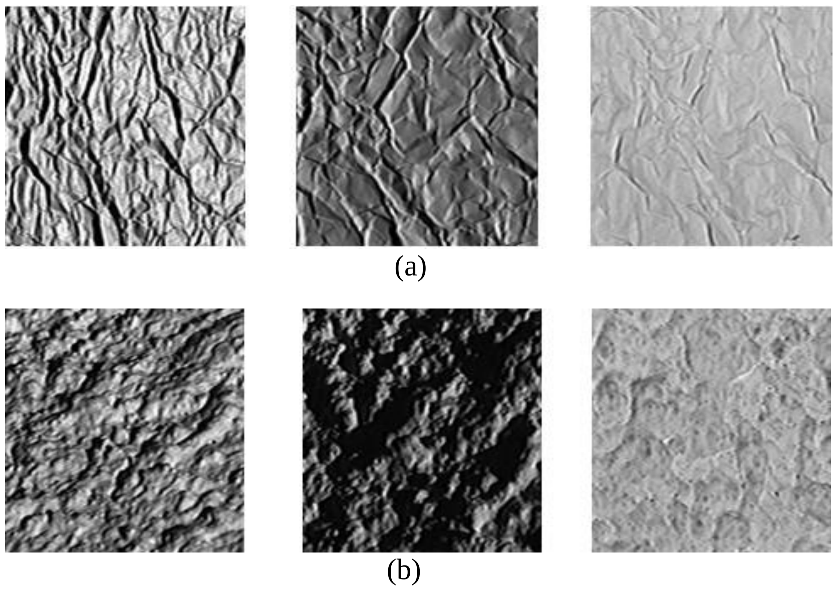


Figure 3.3: Example images from two different classes under different viewpoints and illuminations.

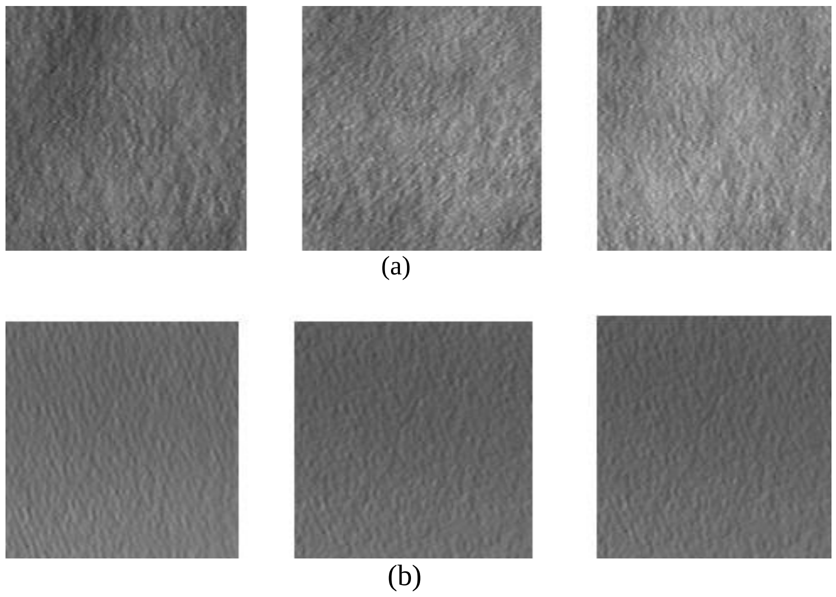


Figure 3.4: Two different classes of texture samples with similar appearance.

A drawback of the CUReT dataset is that images in this dataset have no significant scale variations. Hence, the KTH_TIPS dataset was established to supplement the CUReT dataset by providing a range of scale variations. The KTH_TIPS dataset contains 10 classes of materials which are presented in the CUReT dataset. Textures in each class are imaged at 9 different distances. At each distance, the images are captured under 3 different directions of illumination and 3 different viewpoints. For each class, therefore, 81 texture images are provided. We follow Zhang *et al.* [48] to treat it as a stand-alone dataset. Figs. 3.5 (a) and (b) show some texture images from two different classes captured with large scale variations.

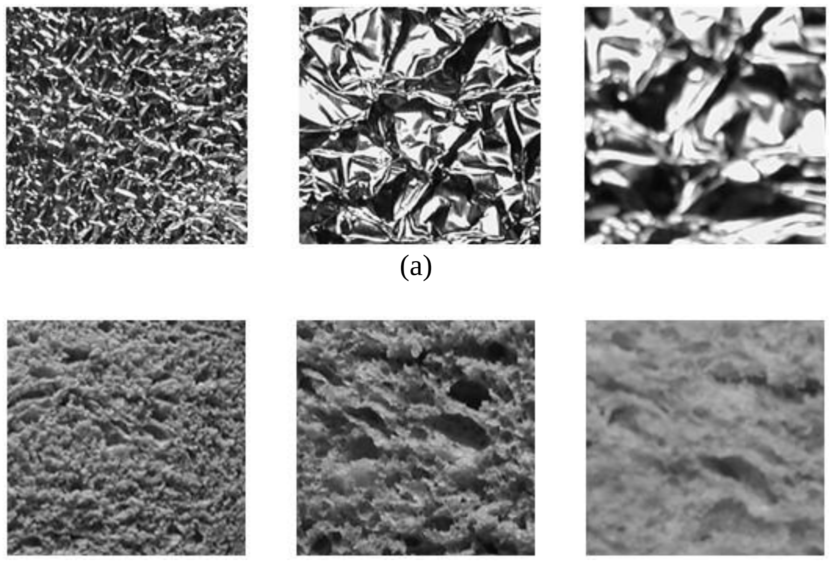


Figure 3.5: Example images from KTH_TIPS with large scale variations.

The evaluation methodology is as follows: M images are chosen per class for training and the remaining images per class are used for testing. In the experiments, for the CUReT dataset, we choose 6, 12, 23, and 46 texture images per class randomly as the training set. For the KTH_TIPS dataset, 5, 10, 20, 30 and 40 images per class are chosen randomly as the training set.

We denote by VZ_MR8 [50] and VZ_Patch [52] Varma and Zisserman's method using the MR8 feature and patch vector feature, respectively. Our proposed methods based on the MR8 and patch vector features are denoted by SR_MR8 and SR_Patch, respectively. For the patch vector feature, a 9×9 neighborhood around each pixel is taken and thus an 81 dimensional feature vector is formed.

Tables 3.1 (a)~(d) and Tables 3.2 (a)~(e) compare our proposed method with VZ_MR8 and VZ_Patch on the CUReT and KTH_TIPS datasets with different training samples and different numbers of textons. From these tables, we can have the following findings. 1) First, the classification accuracies with our proposed method are higher than those with VZ_MR8 and VZ_Patch, respectively. For example, when $M = 6$ on the CUReT dataset, the proposed method with the MR8 feature achieves the accuracies of 81.26%, 81.35% and 81.42% using 20, 30 and 40 textons per class, respectively, while the VZ_MR8 method achieves the accuracies of 80.55%, 80.62% and 80.67%. With the patch vector feature, our proposed method can achieve the accuracies of 81.47%, 81.56% and 81.66% using 20, 30 and 40 textons per class, while the VZ_Patch method can achieve 80.98%, 81.06% and 81.16%. With the increase of the number of training samples, the method can achieve higher accuracies. 2) Second, since the dimension of the patch vector is higher than that of the MR8 filter bank, the classification accuracy with the 81-dimensioanl patch vector is slightly superior to that with the 8-dimensional filter response. 3) Finally, we can observe that the number of textons learned per class has little effect on the final classification accuracy.

We also compare the proposed method to some classical texture classification methods such as LBP [17] and CLBP [18] on the CUReT and KTH_TIPS datasets with different numbers of training samples. The classification accuracies on the two datasets are listed in Table 3.3 and Table 3.4. From these tables, one can see that the classification accuracy with the SR based texton learning method is higher than the classical LBP and state-of-the-art CLBP methods.

Table 3.1: Classification accuracies on the CUReT texture dataset using (a) 6; (b) 12; (c) 23 and (d) 46 training samples.

Textons per class	20	30	40
VZ_MR8 [50]	80.55%	80.62%	80.67%
VZ_Patch [52]	80.98%	81.06%	81.16%
SR_MR8	81.26%	81.35%	81.42%
SR_Patch	81.47%	81.56%	81.66%

(a)

Textons per class	20	30	40
VZ_MR8 [50]	88.65%	88.78%	89.2%
VZ_Patch [52]	89.57%	89.68%	89.73%
SR_MR8	89.77%	89.81%	89.83%
SR_Patch	89.97%	90.06%	90.12%

(b)

Textons per class	20	30	40
VZ_MR8[50]	93.89%	94.01%	94.22%
VZ_Patch[52]	94.43%	94.56%	94.64%
SR_MR8	94.49%	94.58%	94.65%
SR_Patch	94.74%	94.81%	94.95%

(c)

Textons per class	20	30	40
VZ_MR8[50]	96.9%	97.11%	97.28%
VZ_Patch[52]	97.31%	97.48%	97.61%
SR_MR8	97.5%	97.58%	97.62%
SR_Patch	97.62%	97.74%	97.88%

(d)

Table 3.2: Classification accuracies on the KTH_TIPS texture dataset using (a) 5; (b) 10; (c) 20; (d) 30 and (e) 40 training samples.

Textons per class	20	30	40
VZ_MR8 [50]	90.25%	90.42%	90.62%
VZ_Patch [52]	90.38%	90.66%	90.76%
SR_MR8	90.66%	90.75%	90.79%
SR_Patch	90.76%	90.84%	90.96%

(a)

Textons per class	20	30	40
VZ_MR8 [50]	94.15%	94.31%	94.41%
VZ_Patch [52]	94.33%	94.56%	94.68%
SR_MR8	94.67%	94.80%	94.83%
SR_Patch	94.77%	94.96%	95.08%

(b)

Textons per class	20	30	40
VZ_MR8[50]	94.89%	95.01%	95.14%
VZ_Patch[52]	94.96%	95.06%	95.34%
SR_MR8	95.10%	95.32%	95.5%
SR_Patch	95.24%	95.41%	95.75%

(c)

Textons per class	20	30	40
VZ_MR8[50]	95.03%	95.11%	95.20%
VZ_Patch[52]	95.31%	95.48%	95.61%
SR_MR8	95.5%	95.79%	95.82%
SR_Patch	95.62%	95.84%	95.98%

(d)

Textons per class	20	30	40
VZ_MR8[50]	95.53%	95.71%	95.80%
VZ_Patch[52]	95.61%	95.88%	95.91%
SR_MR8	95.95%	96.04%	96.12%
SR_Patch	96.12%	96.24%	96.38%

(e)

Table 3.3: Classification rates on the CUReT dataset by different methods with different numbers of training samples.

Training samples	6	12	23	46
LBP [17]	73.74%	83.78%	90.02%	94.56%
CLBP [18]	73.39%	83.09%	89.15%	93.46%
VZ_MR8 [50]	80.67%	89.2%	94.22%	97.28%
VZ_Patch [52]	81.16%	89.73%	94.64%	97.61%
SR_MR8	81.42%	89.83%	94.65%	97.62%
SR_Patch	81.66%	90.12%	94.95%	97.88%

Table 3.4: Classification rates on the KTH_TIPS dataset by different methods with different numbers of training samples.

Training samples	5	10	20	30	40
LBP [17]	66.60%	77.53%	86.42%	91.16%	93.60%
CLBP [18]	73.35%	83.40%	90.06%	93.25%	95.77%
VZ_MR8 [50]	90.62%	94.41%	95.14%	95.20%	95.80%
VZ_Patch [52]	90.76%	94.68%	95.34%	95.61%	95.91%
SR_MR8	90.79%	94.83%	95.5%	95.82%	96.12%
SR_Patch	90.96%	95.08%	95.75%	95.98%	96.38%

3.5 Summary

In this chapter, we proposed to use the sparse representation (SR) technique to learn the texton dictionary for texture image representation. Consequently, a histogram of SR coefficients was constructed for texture classification. Our experimental results on the CUReT and KTH_TIPS texture datasets validated that the proposed method can achieve the higher classification accuracy than the scheme using the *K*-means clustering method. Moreover, the proposed method is also superior to the classical LBP and state-of-the-art CLBP method.

Chapter 4. Effective Texture Classification by Texton Encoding Induced Statistical Features

In this chapter, we present an effective and efficient texton encoding scheme for texture classification. Section 4.1 introduces briefly the related works. Section 4.2 presents in detail the proposed texton encoding based texture classification scheme, which mainly consists of four stages: texton dictionary learning, texton encoding, feature description and classification. Section 4.3 performs the experiments on three representative and benchmark texture datasets and Section 4.4 presents our conclusions.

4.1 Introduction

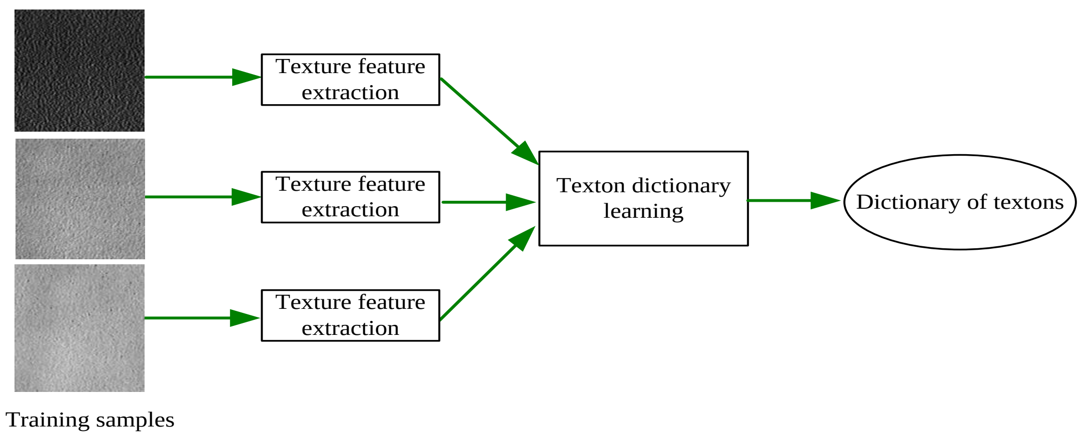
In general, texture classification by textons consists of two main components: texton learning and texture representation by the learned textons. Usually, texture features are first extracted from the images, for example, by MR8 filters [50] or the patch based vectors [52]. Then textons are learned from the feature maps of training texture images. Finally, for a test texture image, we encode its feature map over the learned texton dictionary, and the statistical texton histogram is calculated to represent the test texture for classification.

Plenty of texture classification techniques [41, 42, 43, 44, 47, 48, 50, 52, 55, 56] can be categorized as texton learning based methods. In these methods, a dictionary of textons is usually learned by the *K*-means clustering algorithm, and then the distributions of textons are extracted as statistical features for classification. By the *K*-means clustering algorithm, however, the learned *K* ball-like clusters may not be able to characterize well the intricate feature space of the texture images. For example, for an input vector which lies in the boundary of two or more clusters, the *K*-means clustering will randomly assign it to one of the classes. Recently, the theory of sparse representation (or sparse coding) has been successfully used in various image analysis applications, and sparse coding based texton learning has also been proposed for texture classification in Chapter 3 of this thesis. However, the l_0-norm or l_1-norm minimization in texton learning and encoding makes sparse coding based texture classification very time-consuming. On the other hand, using only the distribution of texton coding coefficients for texture classification makes these methods less effective and robust, particularly when the number of training samples is insufficient.

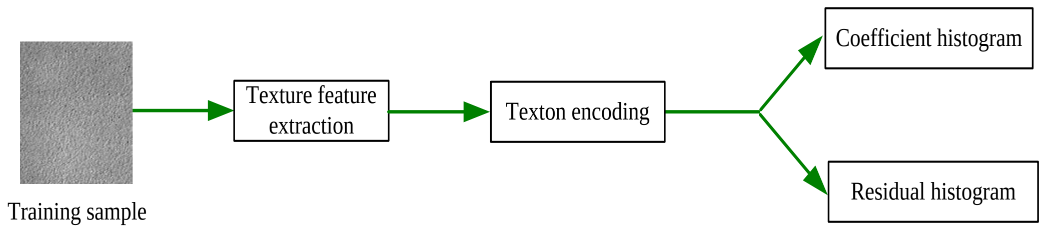
Different from the above texton learning based texture classification methods, in this Chapter we propose a regularized least square based texton learning model, which is much more accurate than the *K*-means clustering while being much more efficient than sparse coding to implement. Meanwhile, we propose a fast texton encoding method to code the texture feature over the learned dictionary, and two types of texton encoding induced statistical features, coefficient histogram and residual histogram, are defined. Finally, the nearest neighbor classifier and the nearest subspace classifier are applied to classify the texture images.

4.2 Texton Encoding Induced Statistical Features for Texture Classification

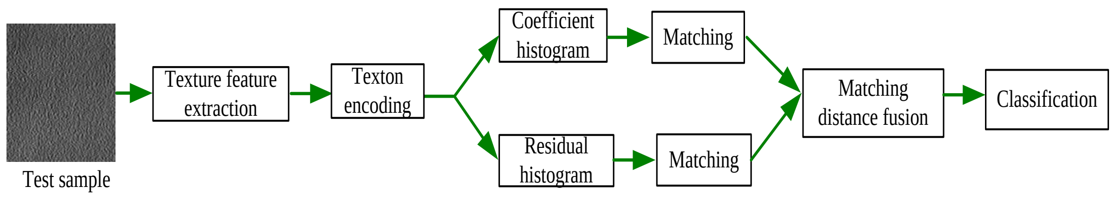
In this section, we describe in detail the proposed texture classification approach, which mainly consists of four stages: texton dictionary learning, texton encoding, feature description and classification. The process of our texture classification method is illustrated in Fig. 4.1.


Texture feature extraction
Texture feature extraction
Texture feature extraction
Texton dictionary learning
Dictionary of textons
Training samples

(a) Texton dictionary learning



(b) Texton encoding and feature description for the training sample



(c) Classification

Figure 4.1: Flow chart of our method: texton dictionary learning, texton encoding, feature description and classification.

4.2.1 Texton Dictionary Learning

In texton based texture classification, the dictionary of textons is learned from the features of training texture images. Before learning the textons, the training texture samples are converted to grey level images and are normalized to have zero mean and unit standard deviation. This process offers a certain amount of invariance to illumination changes. By using some feature extractor (e.g., MR8 filtering [50]), the feature vectors are then extracted from the texture images and normalized by using the Weber's law [50]. In this way, for each class of texture images, we can construct a training dataset $\boldsymbol{X} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n] \in R^{m \times n}$, where $\boldsymbol{x}_i$, $i = 1,2,...,n$, is an m-dimensional feature vector extracted at some location of a training image of this class. In other word, $\boldsymbol{X}$ is the collection of feature vectors extracted from all the samples of one given class. Then the dictionary of textons, denoted by $\boldsymbol{D} = [\boldsymbol{d}_1, \boldsymbol{d}_2, \ldots, \boldsymbol{d}_l] \in R^{m \times l}$, is to be learned from the training dataset $\boldsymbol{X}$, where $\boldsymbol{d}_j$, $j = 1,2,...,l$, is a texton. The dictionary $\boldsymbol{D}$ will be much more compact than $\boldsymbol{X}$, i.e., $l << n$, but it is expected that all the samples in $\boldsymbol{X}$ could be well represented by the learned dictionary $\boldsymbol{D}$.

In many previous texton based texture classification methods [41, 42, 43, 44, 47, 48, 50, 52, 55, 56], the *K*-means clustering algorithm is used to learn the dictionary $\boldsymbol{D}$. 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 in that cluster. 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 Chapter 3 we used the sparse representation model to train such a dictionary by solving $\min_{\boldsymbol{D},\boldsymbol{\Lambda}} \left(\|\boldsymbol{X} - \boldsymbol{D}\boldsymbol{\Lambda}\|_F^2 + \lambda |\boldsymbol{\Lambda}|_1 \right)$, where λ is a positive constant to balance the representation error and sparsity of the coding coefficients. However, the sparse coding is rather time consuming. Furthermore, using sparse representation to learn the dictionary of textons often implies that we have to use sparse coding to encode the query sample, making the texture classification process expensive and slow.

In order for representation accuracy as well as efficiency, in this section we propose a new texton learning model as follows:

$$\min_{\boldsymbol{D},\boldsymbol{\Lambda}}\left(\|\boldsymbol{X}-\boldsymbol{D}\boldsymbol{\Lambda}\|_F^2+\lambda\sum\nolimits_{i=1}^{n}\|\boldsymbol{\alpha}_i\|_2^2+\gamma\sum\nolimits_{i=1}^{n}\|\boldsymbol{\alpha}_i-\boldsymbol{\mu}\|_2^2\right) s.t\ \boldsymbol{d}_j^T\boldsymbol{d}_j=1 \qquad (4\text{-}1)$$

where $\boldsymbol{\Lambda}=[\boldsymbol{\alpha}_1,\boldsymbol{\alpha}_2,\ldots,\boldsymbol{\alpha}_l]\in R^{l\times n}$ is the coding matrix of $\boldsymbol{X}$ over $\boldsymbol{D}$ and $\boldsymbol{\alpha}_i$, $i=1,2,...,n$, is the l-dimensional coding vector of $\boldsymbol{x}_i$; $\boldsymbol{\mu}$ is the mean of all $\boldsymbol{\alpha}_i$, i.e., $\boldsymbol{\mu}=\frac{1}{n}\sum\nolimits_{i=1}^{n}\boldsymbol{\alpha}_i$; 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. (4-1), we use the l_2-norm, instead of the l_1-norm, to characterize the coding vectors based on two considerations. First of all, this will reduce greatly the time complexity of minimization. Second, unlike the dictionary learning for natural image reconstruction [65, 66, 67, 68], where the dictionary is often very over-complete and thus the l_0-sparsity or l_1-sparsity must be imposed on the coding vectors to ensure that the learned dictionary atoms are able to discriminate the many different local structures in natural images, in our application, however, we learn the texton dictionary $\boldsymbol{D}$ class by class, and the variation of texture samples $\boldsymbol{X}$ for each class usually is not so large as that for natural images; therefore, we could relax the strong l_1-norm regularization on $\boldsymbol{\alpha}_i$ to l_2-norm regularization. In addition, considering that the training samples $\boldsymbol{x}_i$ are from the same class of texture pattern and should share certain similarity, in model Eq. (4-1) we enforce their coding vectors $\boldsymbol{\alpha}_i$ to approach their mean $\boldsymbol{\mu}$, i.e., minimizing $\sum_{i=1}^{n}\|\boldsymbol{\alpha}_i-\boldsymbol{\mu}\|_2^2$. This constraint is basically to reduce the intra-class variation for more accurate classification.

The optimization of Eq. (4-1) can be easily conducted by alternatively optimizing $\boldsymbol{D}$ and $\boldsymbol{\Lambda}$. With some random initialization of $\boldsymbol{D}$, we could first fix $\boldsymbol{D}$ to update $\boldsymbol{\Lambda}$, and the problem in Eq. (4-1) is reduced to a regularized least square problem:

$$\begin{aligned}\{\hat{\boldsymbol{\alpha}}_i\} &= \operatorname{argmin}_{\Lambda}\|\boldsymbol{X}-\boldsymbol{D\Lambda}\|_F^2+\lambda\sum_{i=1}^{n}\|\boldsymbol{\alpha}_i\|_2^2+\gamma\sum_{i=1}^{n}\|\boldsymbol{\alpha}_i-\boldsymbol{\mu}\|_2^2\\ &= \operatorname{argmin}_{\{\alpha_i\}}\|\boldsymbol{X}-\boldsymbol{D\Lambda}\|_F^2+\lambda\sum_{i=1}^{n}\|\boldsymbol{\alpha}_i\|_2^2+\gamma\sum_{i=1}^{n}<\boldsymbol{\alpha}_i-\boldsymbol{\mu},\boldsymbol{\alpha}_i-\boldsymbol{\mu}>\\ &= \operatorname{argmin}_{\{\alpha_i\}}\|\boldsymbol{X}-\boldsymbol{D\Lambda}\|_F^2+\lambda\sum_{i=1}^{n}\|\boldsymbol{\alpha}_i\|_2^2+\gamma\sum_{i=1}^{n}(<\boldsymbol{\alpha}_i,\boldsymbol{\alpha}_i>-2<\boldsymbol{\alpha}_i,\boldsymbol{\mu}>+<\boldsymbol{\mu},\boldsymbol{\mu}>)\\ &= \operatorname{argmin}_{\{\alpha_i\}}\|\boldsymbol{X}-\boldsymbol{D\Lambda}\|_F^2+\lambda\sum_{i=1}^{n}\|\boldsymbol{\alpha}_i\|_2^2+\gamma(\sum_{i=1}^{n}\|\boldsymbol{\alpha}_i\|_2^2-n\|\boldsymbol{\mu}\|_2^2)\end{aligned} \quad (4\text{-}2)$$

Let the partial derivative of Eq. (4-2) with respect to $\boldsymbol{\alpha}_i$ equal to 0, we have:

$$\left(\boldsymbol{D}^T\boldsymbol{D}+\lambda\boldsymbol{I}+\gamma\boldsymbol{I}\right)\boldsymbol{\alpha}_i - \boldsymbol{D}^T\boldsymbol{x}_i=\frac{\gamma}{n}\sum_{i=1}^{n}\boldsymbol{\alpha}_i \quad (4\text{-}3)$$

Summing up Eq. (4-3) from $i=1$ to *n*, we have:

$$\sum_{i=1}^{n}\boldsymbol{\alpha}_i=\left(\boldsymbol{D}^T\boldsymbol{D}+\lambda\boldsymbol{I}\right)^{-1}\boldsymbol{D}^T\sum_{i=1}^{n}\boldsymbol{x}_i \quad (4\text{-}4)$$

Substituting Eq. (4-4) into (4-3), it is easy to derive that each coding vector $\boldsymbol{\alpha}_i$ in $\boldsymbol{\Lambda}$ can be analytically solved by:

$$\hat{\boldsymbol{\alpha}}_i=\left(\boldsymbol{D}^T\boldsymbol{D}+\lambda\boldsymbol{I}+\gamma\boldsymbol{I}\right)^{-1}\left(\boldsymbol{D}^T\boldsymbol{x}_i+\gamma(\boldsymbol{D}^T\boldsymbol{D}+\lambda\boldsymbol{I})^{-1}\boldsymbol{D}^T\tfrac{1}{n}\sum_{i=1}^{n}\boldsymbol{x}_i\right) \quad (4\text{-}5)$$

Once all the coding vectors are updated, we could fix $\boldsymbol{\Lambda}$ to update $\boldsymbol{D}$. The objective function in Eq. (4-1) is now reduced to $\min_{\boldsymbol{D}} \|\boldsymbol{X}-\boldsymbol{D}\boldsymbol{\Lambda}\|_F^2$ $s.t.$ $\boldsymbol{d}_j^T\boldsymbol{d}_j=1$. We can update the textons $\boldsymbol{d}_j$ one by one. When update $\boldsymbol{d}_j$, all 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:

$$\hat{\boldsymbol{d}}_j = \operatorname{argmin}_{\boldsymbol{d}_j} \left\|\boldsymbol{X}-\sum\nolimits_{k\neq j}\boldsymbol{d}_k\boldsymbol{\beta}_k-\boldsymbol{d}_j\boldsymbol{\beta}_j\right\|_F^2 \ s.t.\ \boldsymbol{d}_j^T\boldsymbol{d}_j=1 \tag{4-6}$$

Let $\boldsymbol{Y}=\boldsymbol{X}-\sum\nolimits_{k\neq j}\boldsymbol{d}_k\boldsymbol{\beta}_k$, the above equation is:

$$\hat{\boldsymbol{d}}_j = \operatorname{argmin}_{\boldsymbol{d}_j} \left\|\boldsymbol{Y}-\boldsymbol{d}_j\boldsymbol{\beta}_j\right\|_F^2 \ s.t.\ \boldsymbol{d}_j^T\boldsymbol{d}_j=1 \tag{4-7}$$

By the Langrage multiplier, we can obtain:

$$\begin{aligned}\hat{\boldsymbol{d}}_j &= \operatorname{argmin}_{\boldsymbol{d}_j} tr((\boldsymbol{Y}-\boldsymbol{d}_j\boldsymbol{\beta}_j)(\boldsymbol{Y}-\boldsymbol{d}_j\boldsymbol{\beta}_j)^T-\gamma\boldsymbol{d}_j\boldsymbol{d}_j^T-\gamma)\\ &= \operatorname{argmin}_{\boldsymbol{d}_j} tr(-\boldsymbol{Y}\boldsymbol{\beta}_j^T\boldsymbol{d}_j^T-\boldsymbol{d}_j\boldsymbol{\beta}_jY^T+\boldsymbol{d}_j(\boldsymbol{\beta}_j\boldsymbol{\beta}_j^T-\gamma)\boldsymbol{d}_j^T)\end{aligned} \tag{4-8}$$

Let the partial derivative of Eq. (4-8) with respect to $\boldsymbol{d}_j$ equal to 0, we have:

$$-\boldsymbol{Y}(\boldsymbol{\beta}_j)^T-\boldsymbol{Y}(\boldsymbol{\beta}_j)^T+2\boldsymbol{d}_j(\boldsymbol{\beta}_j(\boldsymbol{\beta}_j)^T-\gamma)=0 \tag{4-9}$$

Then we can obtain the solution to $\boldsymbol{d}_j$:

$$\hat{\boldsymbol{d}}_j=\boldsymbol{Y}(\boldsymbol{\beta}_j)^T\left(\boldsymbol{\beta}_j(\boldsymbol{\beta}_j)^T-\gamma\right)^{-1} \tag{4-10}$$

Since $\left(\boldsymbol{\beta}_j(\boldsymbol{\beta}_j)^T-\gamma\right)^{-1}$ is a scalar, and $\boldsymbol{d}_j{}^T\boldsymbol{d}_j=1$, the solution to problem (4-10) is

$$\hat{\boldsymbol{d}}_j=\boldsymbol{Y}\cdot\boldsymbol{\beta}_j^T \Big/ \left\|\boldsymbol{Y}\cdot\boldsymbol{\beta}_j^T\right\|_2 \tag{4-11}$$

and $\gamma=\boldsymbol{\beta}_j(\boldsymbol{\beta}_j)^T-\sqrt{\left\|\boldsymbol{Y}(\boldsymbol{\beta}_j)^T\right\|}$.

Once all the textons $\boldsymbol{d}_j$ in $\boldsymbol{D}$ are updated, we then fix $\boldsymbol{D}$ and updated the coding coefficients $\boldsymbol{\Lambda}$ by using Eq. (4-5), and in turn fix $\boldsymbol{\Lambda}$ and update the dictionary $\boldsymbol{D}$ by Eq. (4-11). Such an alternative optimization process stops till the energy of objective function Eq. (4-1) reaches a local minimum. Finally, the learned texton dictionary $\boldsymbol{D}$ is output. Fig. 4.2 plots the curve of energy of objective function Eq. (4-1) vs. iteration number.

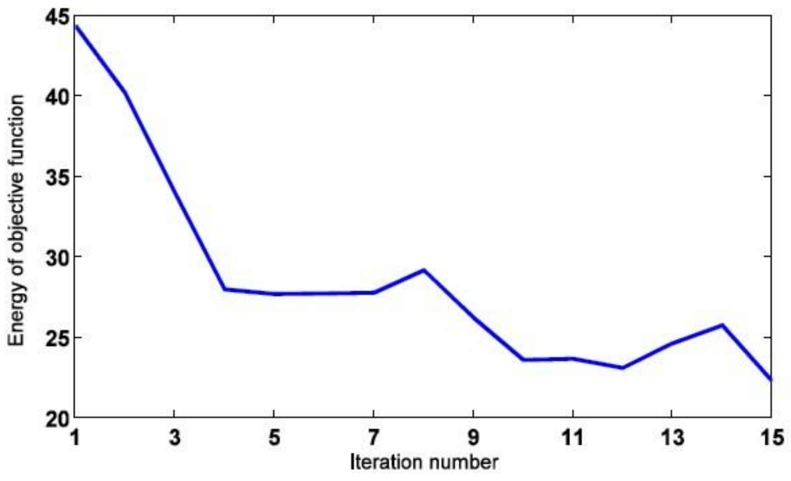


Figure 4.2: Energy of objective function vs. iteration number.

4.2.2 Texton Encoding

By using the texton learning algorithm in Section 4.2.1, for each class k we can learn a texton dictionary $\boldsymbol{D}_k$. Then the dictionaries from all the C classes can be amalgamated into a big texton 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 based texton learning and texture classification methods [41, 42, 43, 44, 47, 48, 50, 52, 55, 56], for each texture feature vector, it is encoded as the label of the texton which is closest to it in dictionary $\boldsymbol{\Phi}$. Then a histogram is built by counting the frequencies of texton labels and used as the statistical feature to describe the texture image.

The texton encoding in the *K*-means based methods [41, 42, 43, 44, 47, 48, 50, 52, 55, 56] is simple but rather coarse. The proposed texton learning model in Eq. (4-1) has much higher representation accuracy than *K*-means by using a few textons to code the feature vector. Obviously, a corresponding texton encoding model needs to be proposed to encode the texture sample. Let's denote by $\boldsymbol{y}_i$ the feature vector extracted at position i of a texture image. One may encode $\boldsymbol{y}_i$ by

$$\hat{\boldsymbol{\alpha}}_i = \arg\min_{\boldsymbol{\alpha}_i} \left(\left\| \boldsymbol{y}_i - \boldsymbol{\Phi}\boldsymbol{\alpha}_i \right\|_2^2 + \lambda \left\| \boldsymbol{\alpha}_i \right\|_2^2 \right) \tag{4-12}$$

which is very fast to optimize because $\hat{\boldsymbol{\alpha}}_i = \boldsymbol{P} \cdot \boldsymbol{y}_i$ and $\boldsymbol{P} = \left(\boldsymbol{\Phi}^T \boldsymbol{\Phi} + \lambda \boldsymbol{I} \right)^{-1} \boldsymbol{\Phi}^T$ can be pre-calculated as a projection matrix. Different from the class-by-class dictionary learning model in Eq. (4-1), where our goal is mainly for the representation power of $\boldsymbol{D}$ for a given class, here $\boldsymbol{\Phi}$ is the amalgamated dictionary of all classes and our goal is a discriminative encoding of $\boldsymbol{y}_i$ for classification, and thus using l_2-norm to regularize $\boldsymbol{\alpha}_i$ is not effective since l_2-norm tends to generate many big coefficients over different classes. Intuitively, using l_1-norm to regularize the encoding is able to generate sparse and more discriminative coding coefficients:

$$\hat{\boldsymbol{\alpha}}_i = \arg\min_{\boldsymbol{\alpha}_i} \left(\left\| \boldsymbol{y}_i - \boldsymbol{\Phi}\boldsymbol{\alpha}_i \right\|_2^2 + \lambda \left\| \boldsymbol{\alpha}_i \right\|_1 \right) \tag{4-13}$$

However, the l_1-minimization is very time consuming, particularly for the amalgamated big dictionary $\boldsymbol{\Phi}$.

Though many fast l_1-minimization techniques have been proposed, such as FISTA [85], ALM [87] and Homotopy [84], it is still desirable if we could find an l_2-minimization based texton encoding method while preserving certain sparsity. In this section we propose such an encoding scheme.

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 most relevant p textons to $\boldsymbol{y}_i$ from $\boldsymbol{\Phi}$ by identifying the p most significant coding coefficients in $|\hat{\boldsymbol{\alpha}}_i|$. Usually we set $p \le m-1$, where m is the dimension of the feature vector $\boldsymbol{y}_i$. Denote by $\boldsymbol{d}_1^i, \boldsymbol{d}_2^i, ..., \boldsymbol{d}_p^i$ the selected p closest textons to $\boldsymbol{y}_i$, and we can then form a small but adaptive sub-dictionary for $\boldsymbol{y}_i$ as $\boldsymbol{\Phi}_i = \left[\boldsymbol{d}_1^i, \boldsymbol{d}_2^i, \cdots, \boldsymbol{d}_p^i\right]$.

Then we encode $\boldsymbol{y}_i$ over the sub-dictionary $\boldsymbol{\Phi}_i$ as:

$$\hat{\boldsymbol{\theta}}_i = \arg\min_{\boldsymbol{\theta}_i} \left(\|\boldsymbol{y}_i - \boldsymbol{\Phi}_i \boldsymbol{\theta}_i\|_2^2 + \lambda \|\boldsymbol{\theta}_i\|_2^2 \right) \tag{4-14}$$

Clearly, this is a simple regularized least square problem as we have $\hat{\boldsymbol{\theta}}_i = \boldsymbol{P}_i \cdot \boldsymbol{y}_i$ and $\boldsymbol{P}_i = \left(\boldsymbol{\Phi}_i^T \boldsymbol{\Phi}_i + \lambda \boldsymbol{I}\right)^{-1} \boldsymbol{\Phi}_i^T$. Since $p \le m-1$ is generally small (please refer to Section 4.3.3 for more information), the computing of $\boldsymbol{\theta}_i$ is very fast, and we can also use the conjugate gradient method to further speed up this encoding process.

The proposed fast two-stage texton encoding scheme can be roughly viewed as a hybrid l_0-l_2-minimization with $\|\boldsymbol{\theta}_i\|_0 \le p$. The first stage selects the p textons and actually sets the coding coefficients over the other textons as 0. This leads to a very sparse representation of $\boldsymbol{y}_i$. The second stage refines the coding coefficients over the selected p textons using the least square method.

4.2.3 Feature Description

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

$$\boldsymbol{\omega}_i(t) = \left|\hat{\boldsymbol{\theta}}_i(t)\right| \Big/ \sum\nolimits_{q=1}^{p} \left|\hat{\boldsymbol{\theta}}_i(q)\right|,\ t = 1,2,...,p \tag{4-15}$$

Since the textons in $\boldsymbol{\Phi}_i$ have a one-to-one correspondence to the textons in $\boldsymbol{\Phi}$, by using $\boldsymbol{\omega}_i$ we can easily re-construct the representation vector of $\boldsymbol{y}_i$ over $\boldsymbol{\Phi}$, denoted by $\boldsymbol{h}_i$, such that

$$\boldsymbol{\Phi}_i \cdot \boldsymbol{h}_i = \boldsymbol{\Phi}_i \cdot \boldsymbol{\omega}_i \tag{4-16}$$

That is, most of the entries in $\boldsymbol{h}_i$ will be 0, and only the entries corresponding to the same textons as those in $\boldsymbol{\Phi}_i$ will have non-zeros values, i.e., $\boldsymbol{\omega}_i$.

Finally, for each feature vector $\boldsymbol{y}_i$ of the texture image, we have a representation vector $\boldsymbol{h}_i$. Then we can form a histogram, denoted by $\boldsymbol{h}_c$, as one statistical description of this texture image:

$$\boldsymbol{h}_c = \sum \boldsymbol{h}_i \tag{4-17}$$

Apart from the encoding coefficient induced histogram, we can also use the encoding residual associated with each of the p selected textons, defined as

$$\boldsymbol{e}_i(t) = \left\| \boldsymbol{y}_i - \boldsymbol{d}_t^i \hat{\boldsymbol{\theta}}_i(t) \right\|_2, \; t = 1,2,...,p \tag{4-18}$$

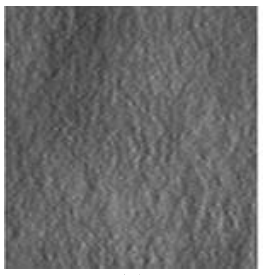
to construct another histogram to describe the texture image. Since the smaller the residual, the more important the associated texton is to represent the feature vector, we normalize $\boldsymbol{e}_i$ into $\boldsymbol{\delta}_i$ by

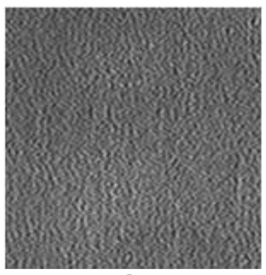
$$\boldsymbol{\delta}_i(t) = \frac{1/\boldsymbol{e}_i(t)}{\sum_{q=1}^{p} 1/\boldsymbol{e}_i(q)} \tag{4-19}$$

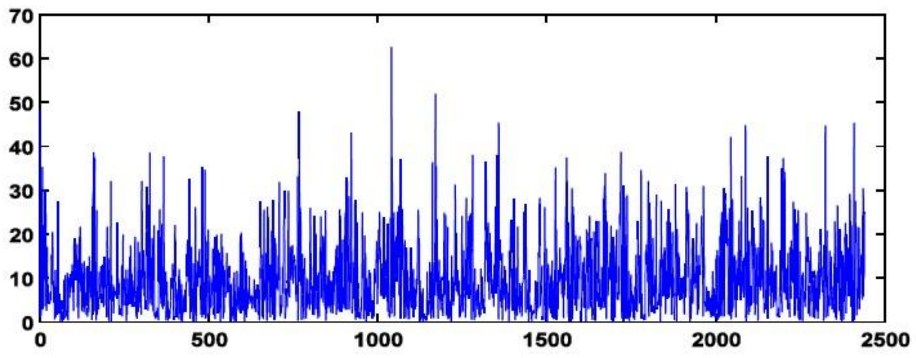
$\boldsymbol{\delta}_i$ is the normalized reciprocal of encoding residual vector associated with $\boldsymbol{\Phi}_i$, and with $\boldsymbol{\delta}_i$ we can readily re-construct the encoding residual reciprocal vector of $\boldsymbol{y}_i$ over $\boldsymbol{\Phi}$, denoted by $\boldsymbol{s}_i$. That 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 the entries in $\boldsymbol{s}_i$ will be 0. Finally, for each feature vector $\boldsymbol{y}_i$ of the texture image, we have an encoding residual vector $\boldsymbol{s}_i$, and then we can then form a histogram, denoted by $\boldsymbol{h}_r$, as another statistical description of the texture image:

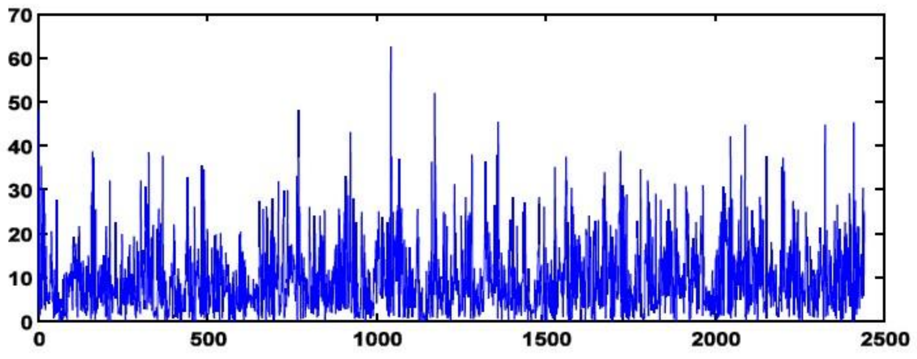
$$\boldsymbol{h}_r = \sum \boldsymbol{s}_i \tag{4-20}$$

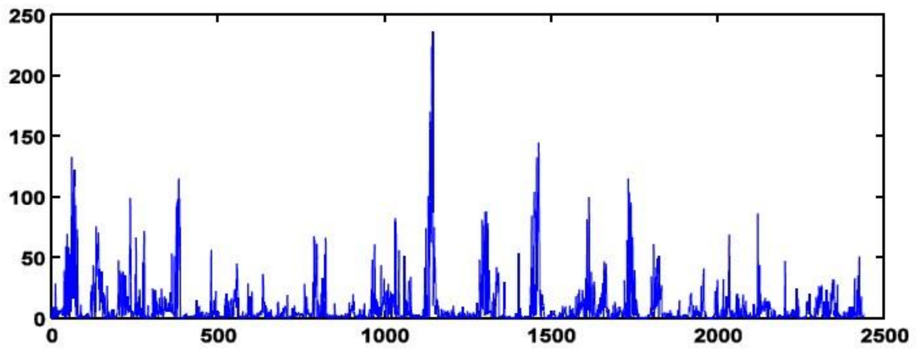
$\boldsymbol{h}_c$ and $\boldsymbol{h}_r$ are two statistical features induced by the encoding coefficients and encoding residual, respectively. Fig. 4.3 shows two texture images of different classes and their histograms. We can see that the histogram features of different classes of textures are very different. Meanwhile, the two types of histograms, $\boldsymbol{h}_c$ and $\boldsymbol{h}_r$, of the same texture image also have enough difference, implying that they have certain amount of complementary information for texture classification.

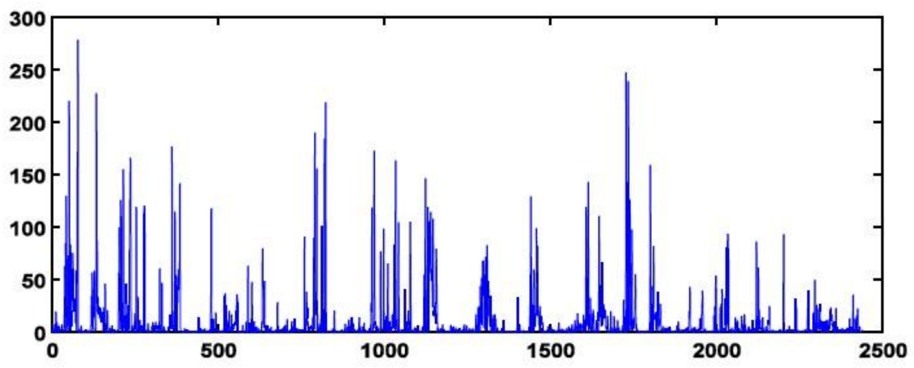

(a)


(b)


70
60
50
40
30
20
10
0
0
500
1000
1500
2000
2500
(c)


70
60
50
40
30
20
10
0
0
500
1000
1500
2000
2500
(d)


250
200
150
100
50
0
0
500
1000
1500
2000
2500
(e)



(f)

Figure 4.3: Two texture images and their histogram features $\boldsymbol{h}_c$ and $\boldsymbol{h}_r$. (c) and (d) are coefficient and residual histogram features of image (a) while (e) and (f) are features of image (b).

4.2.4 Classification

In the stage of classification, we employ the nearest neighbor and nearest subspace classifier to classify textures. For the nearest neighbor classifier (NNC), a test texture image can be classified into the corresponding class by choosing the closest distance between the test sample and the training samples. The χ^2-distance is usually used to compute the distance between two histograms. By NNC, for each of the C classes, we can calculate the shortest χ^2-distance between the query sample $\boldsymbol{y}$ and the training samples of this class by using either the coding coefficient histogram or the coding residual histogram. Then we can have two vectors: $\boldsymbol{\chi}_c^2$ (by coding coefficient histogram) and $\boldsymbol{\chi}_r^2$ (by coding residual histogram), which contain the shortest χ^2-distance from all classes.

To exploit the discriminative information from both coding coefficients and coding residual, we fuse the two distances for classification with the simple weighted average method. In the case of NNC, the fused distance can be represented as:

$$\chi_f^2 = w \cdot \chi_c^2 + (1-w) \cdot \chi_r^2, 0 \le w \le 1 \tag{4-21}$$

where w is the weight. With the fused distance, the query texture image can be classified by:

$$identity(\boldsymbol{y}) = \arg\min_k \chi_f^2(k), k = 1, 2, \cdots, C \tag{4-22}$$

For the nearest subspace classifier (NSC), different from NNC, it can make use of the information of all samples of the same class to compute the distance between the query sample and one class. Suppose that for each of the C classes in the dataset, there are n training samples. Denote by $\boldsymbol{H}_c = [\boldsymbol{h}_{c,1}, \boldsymbol{h}_{c,2}, ..., \boldsymbol{h}_{c,n}]$ and $\boldsymbol{H}_r = [\boldsymbol{h}_{r,1}, \boldsymbol{h}_{r,2}, ..., \boldsymbol{h}_{r,n}]$ the sets of histograms for one class, where $\boldsymbol{h}_{c,i}$ and $\boldsymbol{h}_{r,i}$ are the coding coefficient and coding residual induced histograms of one sample, respectively. For a query texture image ***y***, we first build its two histogram features, denoted by $\boldsymbol{h}_c^y$ and $\boldsymbol{h}_r^y$, and then we can use the NSC to classify ***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, by

$$\boldsymbol{\rho}_c = (\boldsymbol{H}_c^T \boldsymbol{H}_c)^{-1} \boldsymbol{H}_c^T \boldsymbol{h}_c^y \; ; \; \boldsymbol{\rho}_r = (\boldsymbol{H}_r^T \boldsymbol{H}_r)^{-1} \boldsymbol{H}_r^T \boldsymbol{h}_r^y \tag{4-23}$$

and the projection residual can be computed as:

$$err_c = \left\| \boldsymbol{h}_c^y - \boldsymbol{H}_c \boldsymbol{\rho}_c \right\|_2 ; \; err_r = \left\| \boldsymbol{h}_r^y - \boldsymbol{H}_r \boldsymbol{\rho}_r \right\|_2 \tag{4-24}$$

Using Eqs. (4-23) and (4-24), 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 residual from all classes. By the NSC, with $\boldsymbol{err}_*$, $* \in \{c, r\}$, we can classify ***y*** as: $identity(\boldsymbol{y}) = \arg\min_k \boldsymbol{err}_*(k), k = 1, 2, \cdots, C$.

For a more robust classification, we can fuse $\boldsymbol{err}_c$ and $\boldsymbol{err}_r$ as:

$$\boldsymbol{err}_f = w \cdot \boldsymbol{err}_c + (1-w) \cdot \boldsymbol{err}_r \tag{4-25}$$

Then the classification by using the fused information can be done via

$$identity(\boldsymbol{y}) = \arg\min_k \boldsymbol{err}_f(k), k = 1, 2, \cdots, C \tag{4-26}$$

For both Eq. (4-21) and Eq. (4-25), the weight w can be trained from the training dataset with the “leave-one-out” strategy.

4.3 Experimental Evaluation

4.3.1 Methods used in Comparison

In order to evaluate the performance of the proposed method, we compare it with the following state-of-the-art and representative texture classification methods.

LBP [17]: The rotationally invariant uniform local binary pattern (LBP) method.

CLBP [18]: In the completed local binary pattern (CLBP) method, the center pixel of a local region is coded to form a binary code. By the local difference sign-magnitude transformation, the signs and magnitudes are also coded to form a sign and magnitude map to describe the local region. Then the three code maps are combined for texture classification.

VZ_MR8 [50]: Textons are learned by the *K*-means clustering method in MR8 filter response space. Then the texton histogram is formed for classification by labeling each filter response with the texton which is closest to it.

VZ_Patch [52]: Textons are learned by patches in the original image instead of MR8 filter responses.

Hayman *et al.* [90]: Based on the VZ_MR8 method, the SVM classifier is used to classify texture images.

Lazebnik *et al.* [47]: The SPIN and RIFT descriptor are formed on these detected regions by the Harris-affine and Laplacian blob detector to describe textures. The signatures of images can be obtained for classification by clustering these descriptors.

Zhang *et al.* [48]: Based on Lazebnik *et al.*'s method, three descriptors: SIFT, SPIN and RIFT are employed to describe textures and a kernel SVM classifier is used for classification.

Varma and Garg [53]: The local fractal dimension and length from MR8 filter responses are extracted for texture classification.

Crosier and Griffin [88]: Basic image features defined by a partition of a set of six Gaussian derivative filter responses are used as texture features for classification.

Xu *et al.* [21]: The combination of wavelet transformation and MFS [20] are developed for texture classification.

Liu *et al.* [55]: By random projection, random features from patches are extracted. Then textons are learned in the compressed patch domain for classification.

SR based texton learning: The sparse representation based texton learning method we proposed in Chapter 3 is also used in the comparison.

4.3.2 Texture Datasets

Three widely used benchmark texture datasets: CUReT [89], KTH_TIPS [90] and UIUC [47] are used to evaluate the proposed method and competing methods. For the CUReT and KTH_TIPS texture datasets, they were introduced in Chapter 3. Here we introduce the other texture dataset, the UIUC.

The UIUC dataset contains 25 classes, each of 40 images. A major improvement over the CUReT dataset is that there are some significant scale and viewpoint changes as well as some non-rigid deformations in the UIUC dataset. Although there are less severe lighting variations than the CUReT dataset, in terms of intra-class variations in appearance, it is the most challenging one among the commonly used datasets for texture classification. Figs. 4.4(a), (b) and (c) show some example UIUC texture images with significant scale and viewpoint changes as well as deformations.

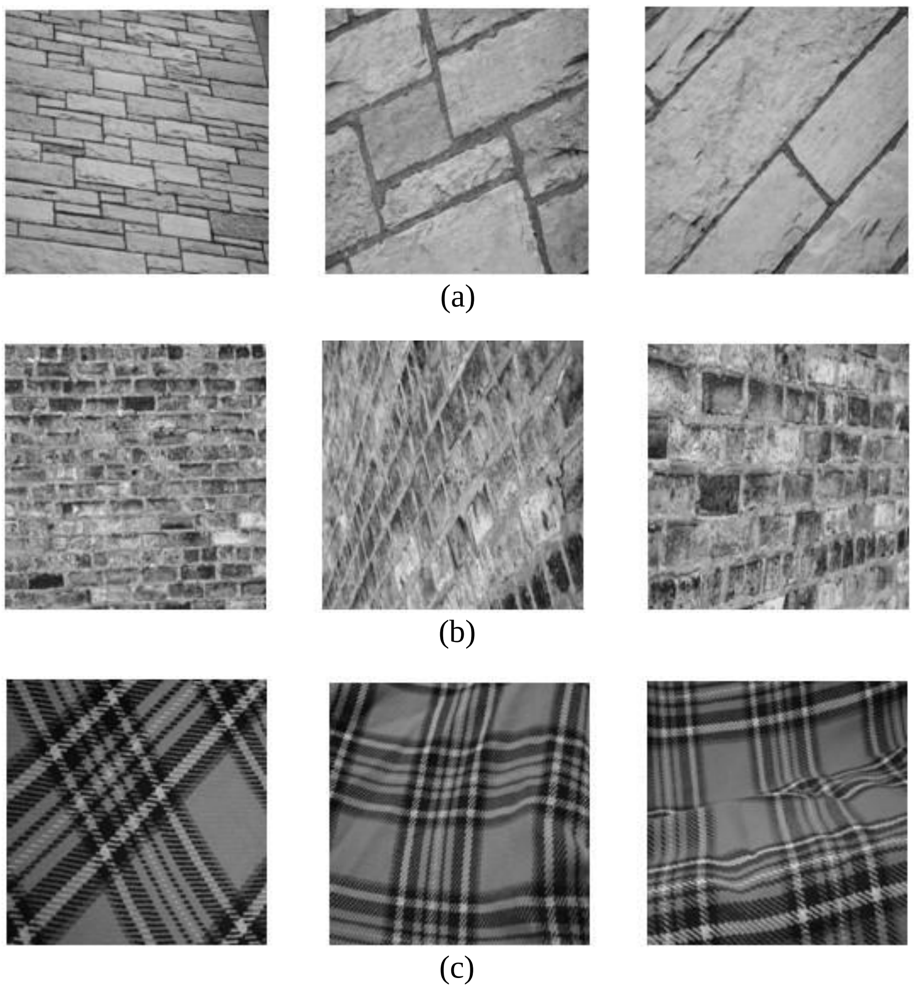


Figure 4.4: Example texture images from the UIUC dataset with some scale and viewpoint changes as well as deformations.

4.3.3 Parameter Selection and Implementation Details

The evaluation methodology on the three datasets is as follows: M images are chosen randomly per class for training and the remaining images are used to form the test set. For CUReT, 6, 12, 23 and 46 images per class are chosen randomly as the training set; for KTH_TIPS, 5, 10, 20, 30 and 40 images per class are randomly chosen as the training set; for UIUC, 5, 10, 15 and 20 training images are chosen randomly per class. For each setting, the experiments were repeated 100 times and the average classification accuracy is reported.

For the CUReT and KTH_TIPS datasets, we employed the MR8 feature [50] as the texture feature 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-stage fast texton encoding we choose $p = 7$ textons to form the sub-dictionary for encoding.

For the UIUC dataset, to address the large scale variation we employ the MR8 feature and SIFT descriptor [46] to represent the texture feature. For each class, $l = 40$ MR8 textons are learned, and $l = 100$ SIFT textons are learned. Then we combine the histogram features to classify 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 feature fusion, the optimal weight w is determined by the leave-one-out method on the training set. With NNC, when 6, 12, 23 and 46 training samples per class are used for the CUReT dataset, the optimal w are 0.65, 0.45, 0.4 and 0.5, respectively; for the KTH_TIPS dataset, when 5, 10, 20, 30 and 40 training samples per class are used, the weights are 0.75, 0.65, 0.65, 0.7 and 0.6, respectively; the optimal weights are 0.6, 0.85, 0.55 and 0.55, respectively, for the UIUC dataset when 5, 10, 15 and 20 randomly chosen training samples are used.

For the feature fusion in NSC based classification, when 6, 12, 23 and 46 training samples per class are used for the CUReT dataset, the optimal w are 0.5, 0.65, 0.4 and 0.5, respectively; for 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; the optimal weights are 0.6, 0.75, 0.75 and 0.55, respectively, for the UIUC dataset when 5, 10, 15 and 20 randomly chosen training samples are used.

4.3.4 Experimental Results

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 features, respectively. As mentioned in 4.3.1, we compare our methods with representative and recently proposed state-of-the-art texture classification methods [17, 18, 21, 47, 48, 50, 52, 53, 55, 88, 90]. Since many competing methods do not have publically released source codes available, we cropped the results from the 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 is used in [17, 18, 47, 50, 52, 53], the SVM classifier is used in [48, 54, 55, 56, 90], etc. For the LBP [17], CLBP [18] and VZ_MR8 [50] methods, we compared them with both the NNC and NSC classifiers.

Table 4.1 compares the state-of-the-art 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). The NSC classifier is used in the LBP, CLBP, VZ_MR8 and the proposed TEISF_c, TEISF_r and TEISF_f methods. One can see that when there are enough training samples, most of the methods can achieve not bad classification accuracy. The method of Lazebnik *et al.* [47] only achieves an accuracy of 72.5% because on the CUReT dataset the combined multiple detectors cannot produce enough regions for a robust statistical characterization of texture. The proposed TEISF_f method achieves the best result on CUReT and UIUC, and it is only slightly worse than Liu *et al*.'s method [55] by a gap of 0.39% on the KTH_TIPS. In this experiment, the gain of TEISF_f over TEISF_c and TEISF_r is not significant because TEISF_c and TEISF_r can already achieve good result and thus no much new information can be introduced in the fusion.

By decreasing the number of training samples per class, in Tables 4.2, 4.3 and 4.4 we present the classification accuracies on the three datasets with the NNC and NSC classifiers, respectively. (Note that not all the competing methods employed in Table 4.1 are reported in Tables 4.2~4 since their results are not available.) From these tables, we can readily make the following findings. 1) First, the performance of the proposed TEISF methods with NSC is superior to that of the proposed method with NNC. 2) Second, the proposed TEISF_f with NSC achieves the best classification accuracy in almost every case. 3) Third, with the decrease of the number of training samples, the advantage of TEISF_f over other methods is getting more and more obvious. 4) Fourth, TEISF_f with NSC 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%. 5) Fifth, when the number of training samples is small, TEISF_c or TEISF_r each 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 residual have complementary information for classification. 6) Sixth, some methods may work well on one dataset but not so well on other datasets. For example, Xu *et al.*'s method [20, 21] has good accuracy on the UIUC dataset, whereas its accuracy is not so good on CUReT. In comparison, the proposed TEISF_f method consistently leads to good results across all the datasets.

At last, in Fig. 4.5 we plot the curves of classification accuracy vs. number of training samples on the three datasets by TEISF_f. The curves by VZ_MR8 [50]+NSC, CLBP [18]+NSC and the recently developed scheme by Liu *et al.* [55] are also plotted for comparison. (Note that the classification accuracies of Liu *et al.*'s method [55] on CUReT with 4 and 8 training samples per class are unavailable.) Clearly, the proposed method is much more robust to the number of training samples.

4.3.5 The Effect of Parameter *p*

In this section, we study the effect of different numbers of the selected textons in the texton encoding stage on classification accuracy. In some sense, the number of selected textons (parameter *p*) can represent the sparsity of coding coefficients in the texton encoding. Figs. 4.6 (a)~(c) show the classification rates under different number of selected textons on the CUReT, KTH_TIPS and UIUC datasets, respectively. Note that two dictionaries (for MR8 and SIFT features) are used for feature extraction on the UIUC dataset. Thus in Fig. 4.6 (c) the x-axis labels the pair of the numbers of selected MR8 and SIFT textons. From those figures, we can observe that when *p* is big (e.g., $p>10$ for the MR8 feature), the sparsity of coding coefficients may be reduced and the classification rate also decreases. When *p* is too small (e.g., $p<5$ for the MR8 feature), the representation is not accurate so that the classification rate is not very good either. When *p* is slightly less than the feature dimensionality (e.g., *p*=7 for the MR8 feature), a good balance of representation accuracy and sparsity is reached so that the best classification rate can be obtained. For example, for the CUReT dataset, with 46 training samples, when 7 textons are chosen, the classification accuracy of 99.54% can be obtained.

4.4 Summary

In this chapter, we proposed a texton encoding based texture classification method, which is simple to implement but shows very promising performance. The textons were learned to ensure the representation accuracy while reducing the within-class variance. A two-stage 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 induced from the texton encoding outputs, namely coding coefficient induced histogram and coding residual induced histogram, were then defined, with which the nearest neighbor classifier and the nearest subspace classifier were applied for texture classification. The experimental results demonstrated that the proposed method outperforms state-of-the-art methods, especially when the number of training samples is small.

Table 4.1: Texture classification rates by representative and state-of-the-art methods when the number of training samples is relatively large.

	CUReT (46)	KTH_TIPS (40)	UIUC (20)
LBP [17]+NSC	99.11%	97.19%	80.3%
CLBP [18]+ NSC	93.46%	96.74%	95.18%
VZ_MR8[50]+NSC	98.17%	97.06%	96.74%
VZ_Patch [52]	98.03%	92.40%	97.83%
Hayman *et al*. [90]	98.46%	94.8%	92%
Lazebnik *et al*. [47]	72.5%	91.13%	93.62%
Zhang *et al*. [48]	95.3%	96.1%	98.7%
Varma and Garg [53]	97.5%	---	95.4%
Crosier and Griffin [88]	98.6%	98.5%	98.8%
Xu *et al*. [21]	---	---	98.60%
Liu *et al*. [55]	99.37%	**99.29%**	98.56%
TEISF_c+NSC	99.03%	97.53%	98.18%
TEISF_r+NSC	99.15%	97.67%	98.22%
TEISF_f+NSC	**99.54%**	98.9%	**99.54%**

Table 4.2 a: Classification rates on the CUReT dataset with different numbers of training samples using the NNC.

Training samples	6	12	23	46
LBP [17]	73.74%	83.78%	90.02%	94.56%
CLBP [18]	73.39%	83.09%	89.15%	93.46%
VZ_MR8 [50]	80.67%	89.02%	94.22%	97.28%
Varma and Garg [53]	81.67%	89.74%	94.69%	97.5%
SR_MR8	81.42%	89.83%	94.65%	97.62%
TEISF_c	81.93%	90.3%	95.0%	97.72%
TEISF_r	83.67%	90.6%	95.15%	97.93%
TEISF_f	85.6%	91.3%	95.34%	98.19%

Table 4.2 b: Classification rates on the CUReT dataset with different numbers of training samples using the NSC.

Training samples	6	12	23	46
LBP [17]	82.25%	91.71%	96.24%	99.11%
CLBP [18]	73.39%	83.09%	89.15%	93.46%
VZ_MR8 [50]	86.33%	92.79%	96.45%	98.17%
Liu *et al* [55]	86.48%	96.43%	97.71%	99.37%
SR_MR8	86.74%	93.43%	96.94%	98.45%
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 4.3 a: Classification rates on the KTH_TIPS dataset with different numbers of training samples using the NNC.

Training samples	5	10	20	30	40
LBP [17]	66.60%	77.53%	86.42%	91.16%	93.60%
CLBP [18]	73.35%	83.40%	90.06%	93.25%	95.77%
VZ_MR8 [50]	90.62%	94.41%	95.14%	95.20%	95.80%
SR_MR8	90.79%	94.83%	95.50%	95.82%	96.12%
TEISF_c	90.89%	94.84%	95.85%	96.20%	96.69%
TEISF_r	91.44%	94.76%	95.31%	96.01%	96.98%
TEISF_f	91.88%	95.32%	96.1%	96.47%	97.21%

Table 4.3 b: Classification rates on the KTH_TIPS dataset with different numbers of training samples using the NSC.

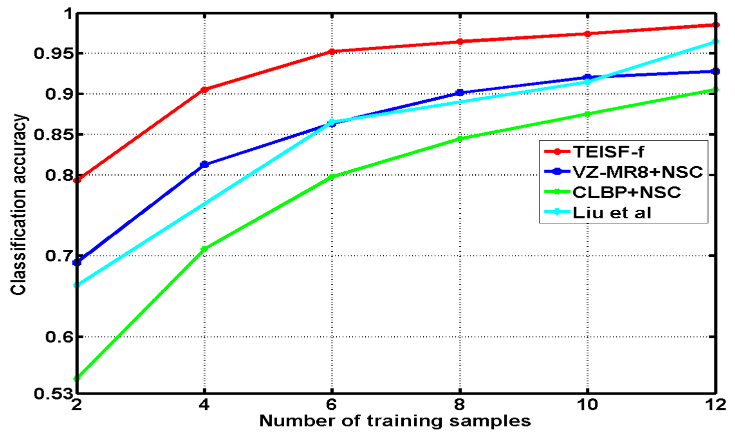
Training samples	5	10	20	30	40
LBP [17]	72.50%	85.41%	94.12%	95.73%	97.19%
CLBP [18]	78.51%	87.69%	94.78%	96.05%	96.74%
VZ_MR8 [50]	91.87%	94.45%	95.32%	95.70%	97.06%
Liu *et al*. [55]	80.90%	89.49%	96.40%	**98.4%**	**99.29%**
SR_MR8	92.03%	94.86%	95.87%	96.1%	97.44%
TEISF_c	92.11%	95.61%	96.99%	97.06%	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 4.4 a: Classification rates on the UIUC dataset with different numbers of training samples using the NNC.

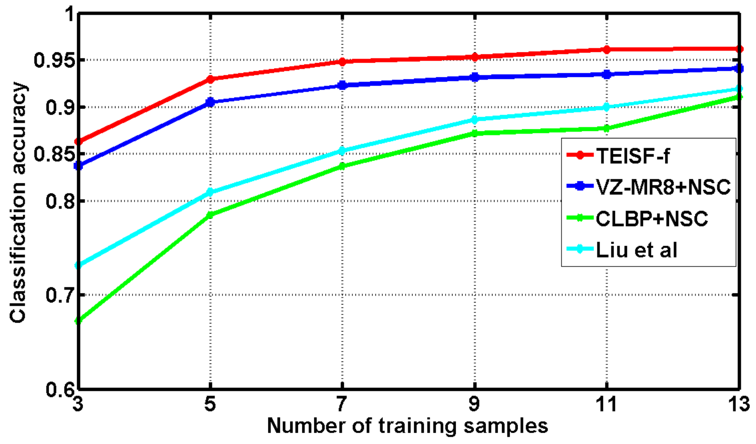
Training samples	5	10	15	20
LBP [17]	38.25%	47.55%	53.70%	57.53%
CLBP [18]	71.22%	81.00%	85.30%	88.13%
VZ_MR8 [50]	82.64%	90.04%	92.04%	93.08%
Lazebnik *et al*. [47]	84.77%	90.17%	92.42%	93.62%
Varma and Garg [53]	85.35%	91.64%	94.09%	95.4%
TEISF_c	84.20%	89.95%	92.55%	94.00%
TEISF_r	86.87%	90.38%	93.08%	94.09%
TEISF_f	87.66%	91.30%	93.34%	94.39%

Table 4.4 b: Classification rates on the UIUC dataset with different numbers of training samples using the NSC.

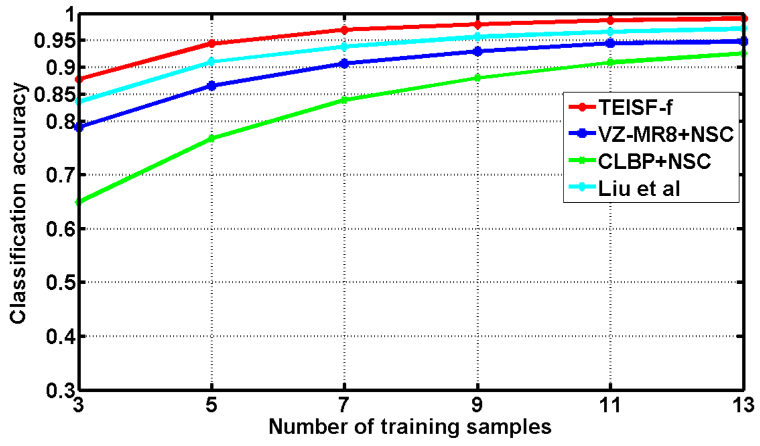
Training samples	5	10	15	20
LBP [17]	43.34%	63.15%	74.15%	80.30%
CLBP [18]	76.73%	89.50%	94.00%	95.18%
VZ_MR8 [50]	86.54%	93.56%	95.71%	96.74%
VZ_Patch [52]	90.17%	95.18%	96.94%	97.83%
Xu *et al*. [21]	93.42%	96.95%	98.01%	98.60%
Liu *et al*. [55]	90.96%	96.00%	97.59%	98.56%
TEISF_c	91.05%	95.30%	97.38%	98.18%
TEISF_r	93.29%	95.82%	97.65%	98.22%
TEISF_f	**94.37%**	**98.38%**	**99.35%**	**99.54%**



(a)

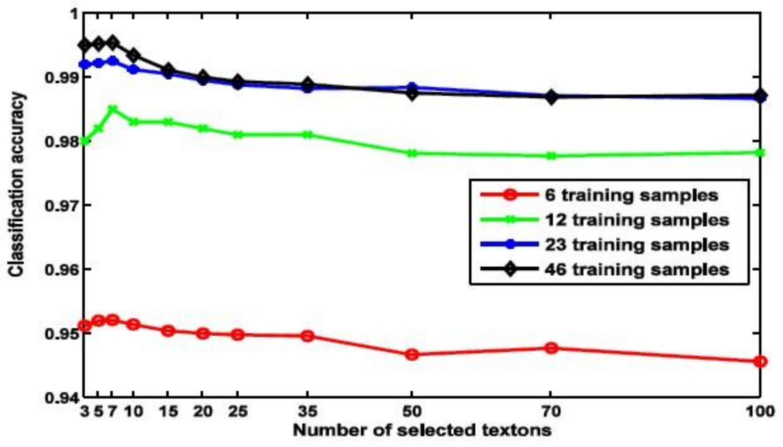


(b)

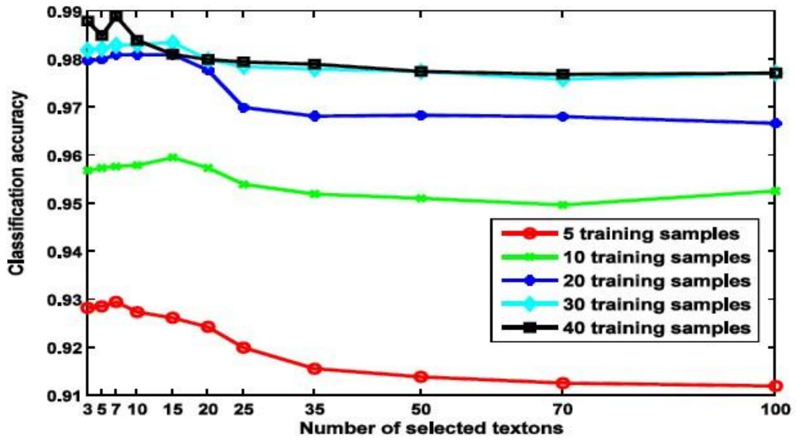


(c)

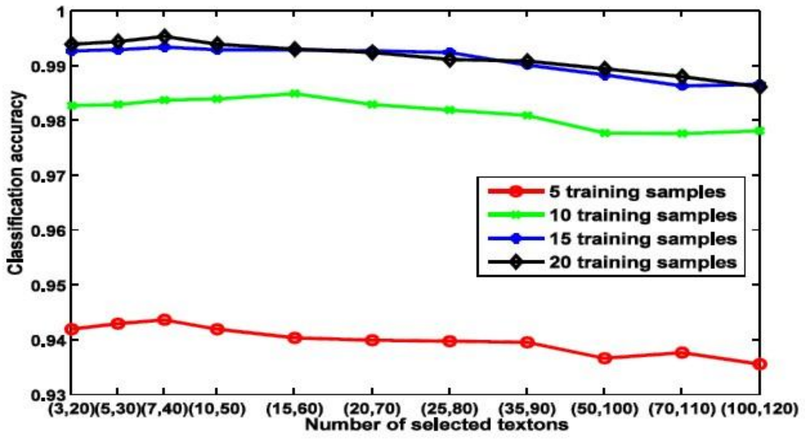
Figure 4.5: The curves of classification rate vs. number of training samples by the proposed TEIFS_f, VZ_MR8 [50]+NSC, CLBP [18]+NSC and the method in [55] on three datasets: (a) CUReT, (b) KTH_TIPS and (c) UIUC.



(a)



(b)



(c)

Figure 4.6: The curves of classification rate vs. number of selected textons on the three datasets: (a) CUReT, (b) KTH_TIPS and (c) UIUC.

Chapter 5. Hand Back Skin Texture Analysis for Personal Identification and Gender Classification

In this chapter, we study the use of hand back skin texture (HBST) for personal identification and gender classification. Section 5.1 briefly introduces some applications with skin texture analysis to biometrics. Section 5.2 introduces the design and architecture of our HBST imaging device. Section 5.3 presents the feature extraction for HBST classification. In Section 5.4, the HBST classification methods are validated on the established HBST dataset for personal identification and gender classification. Section 5.5 concludes the chapter.

5.1 Introduction

Skin, as the outermost part of human body, is known to provide much useful information for health condition analysis [91, 92] and human identity recognition [93], etc. Skin appearance can be viewed as a kind of texture surface, and skin texture analysis can be used in various applications. For example, in [92], skin texture analysis is applied to computer-aided diagnosis in dermatology, where the dermatologist can use the computational texture representation to make an initial diagnosis for the patient. Meanwhile, biomedical evaluation based on skin texture can provide some tests for topical skin treatments, which can be used to judge whether these treatments are effective or not in the early stages. In addition, skin texture analysis can be used to estimate human skin age [94, 95].

With the rapid development of computer techniques, researchers have investigated the use of various biometric traits, including fingerprint [96, 97, 98], face [99, 100], iris [101, 102], retina [103, 104], palmprint [105, 106, 107, 108] and finger knuckle [109], etc, for the purpose of personal authentication. Moreover, face [116, 117] and gait [118] have been used for gender classification. In [117], the authors demonstrated that the SVM classifier is able to learn and classify gender from a set of hairless low resolution face images with high classification accuracy. For gait based gender recognition, a number of combinations of gait components [118] are extracted to classify gender with the SVM classifier. Skin texture, as a potential biometric identifier to assist existing biometric traits, has also received some attention in the past years. Based on the locally consistent property of the fingerprint skin tissue, Rowe [110] extracted texture features of the fingerprint skin for human identification while reducing the size of the fingerprint sensing area. Cula and Dana *et al.* [92, 93] used the bidirectional texture function (BTF), which is analogous to the bidirectional reflectance distribution function (BRDF), to model skin texture to assist face recognition. For each skin texture surface, the bidirectional texture function is sampled in multiple camera views and illumination directions. However, obtaining accurate bidirectional image measurements of skin texture surface is hard, because the skin surface is non-planar, non-rigid and can be stretched.

It can be observed that the human hand back skin has a clear and consistent texture pattern which is uniformly distributed over a large portion of hand back. Based on our daily life experience, we know that the hand based skin texture (HBST) pattern is not permanent and it will change over time. For example, young people will have finer (i.e., smoother and smaller size of micro-cells) HBST than old people, while female will have much finer HBST than male. Nonetheless, in a relatively long period, the HBST of a person is stable. Based on [124], the changes in skin associated with age can be visualized by gloss and wrinkles, and thus some measurements of wrinkles, gloss and density of microgrooves of skin can be used for age estimation. In [124], the number of pixels in the binary image of the epidermal cross-section is used to estimate the age. From the curve of measured peripheral length vs. age in [124], one can see that the peripheral length changes little in 1-2 years, which means that skin texture can keep stable in a relatively long period. These motivate us to investigate the possibility that if the HBST pattern can be used to aid personal identification and gender classification. Many biometric identifiers such as fingerprint, face, iris and palmprint, etc, have been proposed for human identification, and our goal is not to compete with those biometric identifiers, but to validate whether HBST has a certain level of accuracy so that it can be helpful to assist the existing biometric authentication techniques. Moreover, apart from biometric applications, as a specific kind of texture patterns, the established HBST dataset can also be used to evaluate the texture feature extraction and classification algorithms in the community of computer vision and pattern recognition.

In this chapter, we study the use of HBST for personal identification and gender classification. To this end, an HBST imaging device is first designed to capture HBST images. Since HBST is a type of fine scale feature, a high resolution (about 450 dpi) is set to capture the detailed texture patterns in hand back images. Different from the method in [92, 93], where skin texture is modeled as a 3D texture and the BTF is used to describe the skin appearance, we model HBST as a kind of 2D appearance texture because the hand back can be approximately viewed as a 2D plane. Therefore, we directly capture the HBST image using a CCD camera with the fixed position under the fixed illumination direction. Such a design makes the HBST image acquisition very efficient and feasible for the purpose of personal identification and gender classification. In the 3D model [92, 93], multiple cameras and multi-illuminations are needed to collect samples, which makes the imaging system very complex. Compared with the 3D model, modeling the hand back skin surface with the 2D model makes our imaging system much easier to design and more convenient to collect samples. In addition, our goal is to analyze the texture pattern in hand back skin so that 2D modeling is more suitable.

By using the designed HBST imaging device (please refer to Section 5.2 for more details), an HBST image dataset is established, which consists of 1920 images from 80 volunteers (160 hands). The texton learning based methods proposed in Chapters 3 and 4 are then employed for HBST pattern classification. The HBST images are passed through a bank of filters, and a set of textons are learned from the filter responses with the texton learning technique. Then, features with the learned textons are extracted for classification. The performance of the proposed texton learning based texture classification methods is evaluated by using the established HBST dataset in comparison with state-of-the-art texture classification schemes, including the multi-fractal spectrum [20], original LBP [17], dominant LBP [19], completed LBP [18], and the texton learning based method in [50]. Experimental results demonstrated that HBST could achieve interesting personal identification and gender classification accuracy, which implies that HBST can be used to aid the existing biometric authentication techniques to improve the performance. Meanwhile, the established HBST dataset is very challenging, providing a good platform to develop and test texture classification algorithms.

5.2 The HBST Imaging System

The schematic diagram of the major components of the developed hand back skin texture (HBST) imaging system is illustrated in Fig. 5.1. It is composed of a ring of LED light source, a lens and the associated CCD camera, and a data acquisition card. When it works to collect data, the LED light source will illuminate the hand back skin, and then the CCD camera will capture the HBST image and pass it to the data acquisition card. The data acquisition card will then transmit the image to the data processing unit (e.g., the CPU in a PC).

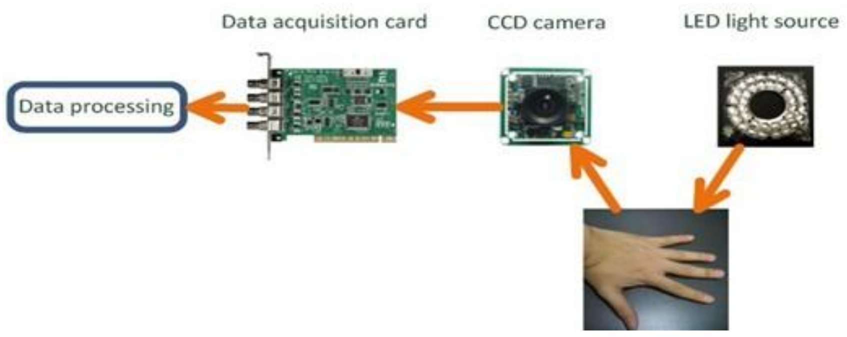


Figure 5.1: The schematic diagram of the developed hand back skin texture imaging system.

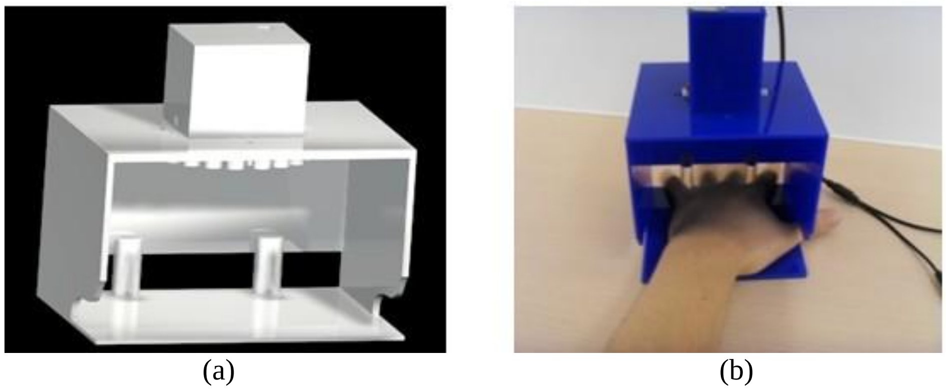


Figure 5.2: (a) The inner structure of the developed hand back skin texture imaging system. (b) The outlook of the imaging system.

Fig. 5.2(a) illustrates the inner structure of the HBST imaging device and Fig. 5.2(b) shows its outlook. A critical issue in HBST data acquisition is to make the data collection environment as stable and consistent as possible so that the undesired disturbance (e.g., the background and environmental illumination disturbances) can be reduced. Meanwhile, a stable data collection environment can effectively reduce the complexity of feature extraction and improve the classification accuracy. Specifically speaking, in our imaging system how to keep the illumination uniform and constant and how to fix the position of the hand are of the most importance. To this end, a ring of LED light source (in visible spectrum, 390nm ~ 780nm) and the CCD camera are enclosed in a box to keep the illumination nearly constant. The LEDs are arranged in a circle around the camera to make the illumination uniform. Refer to Figs. 5.2(a), in order to capture the central part of the hand back skin texture image, two pegs are used to fix the hand, which can guide the position of index and little fingers with a friendly user interface. And this can reduce largely the pose variations of the hand in different capturing sessions. In addition, our design could make the skin texture surface as flat as possible so that we can model the skin surface as a 2D planar texture image.

Note that there are some differences between our device and the palmprint device [106]. First, in order to capture the micro-structures of HBST, the resolution of the chosen camera in our device is higher than that in the palmprint device. Second, the light source is different from that in the palmprint device. In our device, the ring LED is used while the halogen light source is used in the palmprint device. Finally, the architecture of the device is different. In our HBST imaging system, we employ the micro-industrial CCD camera board, LED light source and USB data acquisition card to collect data. However, in the palmprint device, the commonly used industrial CCD camera, halogen light source and PCI data acquisition card are used to collect data.

The texture pattern of human hand back skin can only be clearly observed in a relatively fine scale. In order to capture the HBST image in a high enough resolution while avoiding the HBST image size to be too big, the focal of the lens should be carefully designed. In our imaging system, due to the limited distance between the camera and the hand back, we choose to use a 12mm focal length lens to capture the HBST image. Further reduction in the focal length will distort the captured image. The size of the CCD output image is 576×768 (the raw image is saved in the 24-bitmap format and we convert it into 8-bit gray level image), and finally the HBST image is captured under a resolution of about 450 dpi. In designing our imaging device, we tested different resolution settings of the HBST image, and found that the resolution of about 450 dpi can satisfy our requirements. If the resolution of the image is too low, the micro-structures such as wrinkles in the image cannot be captured clearly. If the resolution of the image is too high, the cost of the camera will be high and the computational cost will also increase. A resolution of 450 dpi is good enough to capture clear HBST images with a low cost.

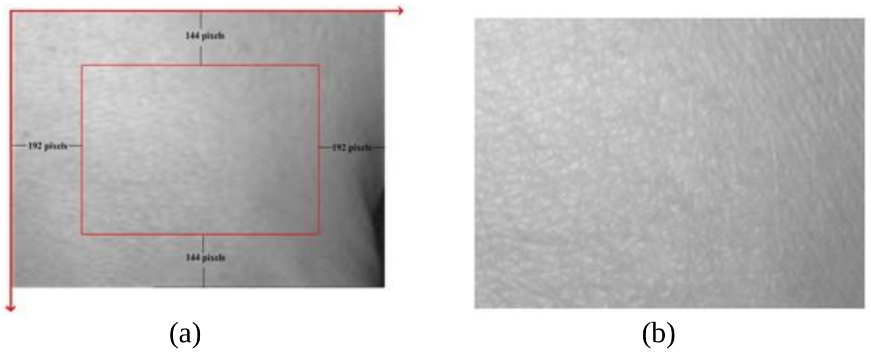


Figure 5.3: (a) is the raw image (size 576×768) captured by our device and (b) is the sub-image (size 288×384) cropped from the central part of (a).

In our HBST imaging system, since we use two pegs to fix the hand position, the top and bottom boundary of the captured skin texture image can be roughly fixed. Although the hand back skin can be viewed nearly as a 2D plane in the central part, the boundary part of the hand back can be much distorted in the captured HBST image. In order to reduce the effect of the hand back boundary area on the later feature extraction and recognition procedures, we can crop a sub-image from the captured raw image by removing the four boundary areas. Refer to Fig. 5.3, we simply set the top left corner of the HBST image as the origin point, and based on our experimental experience we crop the central part of size 288×384 from the original image of size 576×768. Fig. 5.3 illustrates the cropping process. Such a sub-image cropping process can not only make the feature extraction more stable and accurate, but also reduce the computational cost.

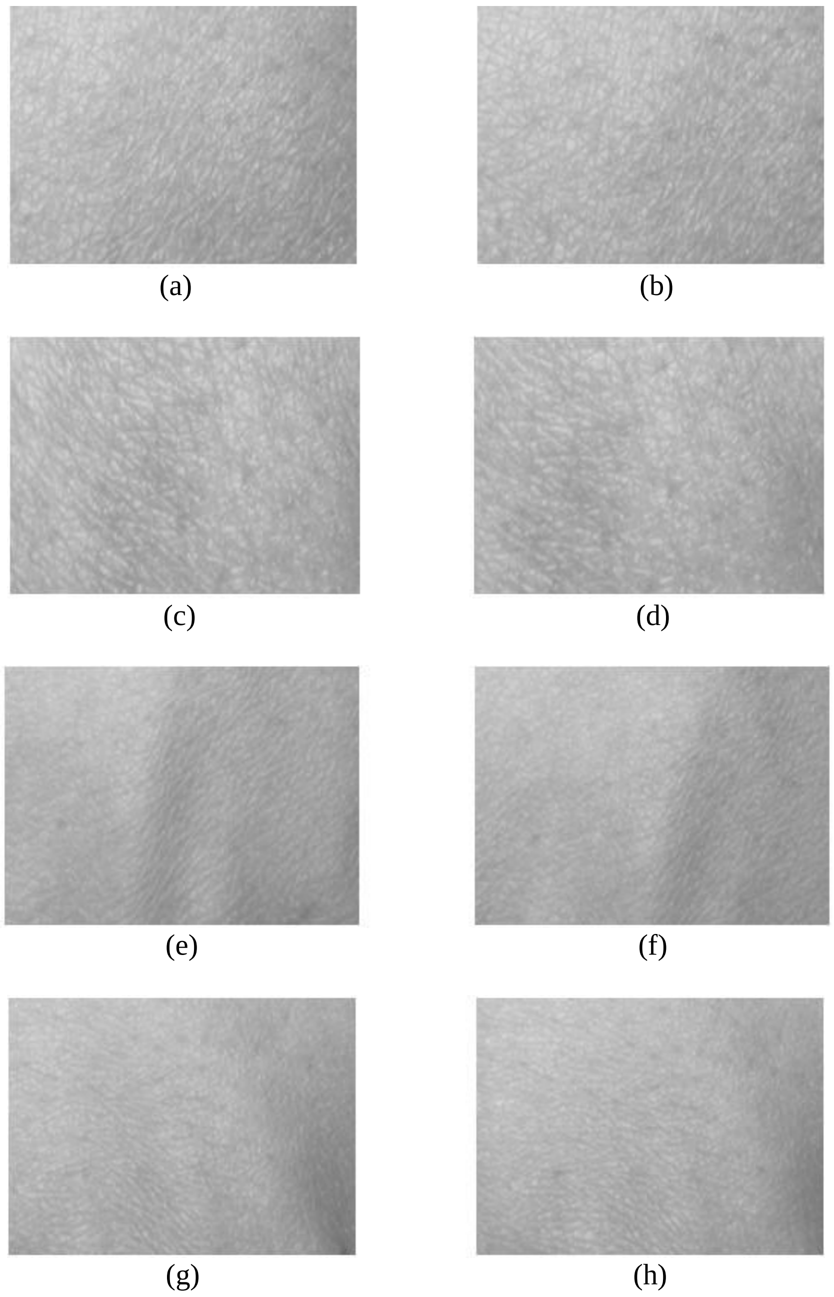


Figure 5.4: (a) and (b) are the cropped left-hand HBST images of a person collected in two different sessions while (c) and (d) are the right-hand HBST images from the same person. (e) and (f) are the cropped left-hand HBST images from another person, while (g) and (h) are the right-hand HBST images from this person.

Fig. 5.4 shows some example cropped HBST images captured in two different sessions with the time interval of about 30 days. Figs. 5.4(a) and 5.4(b) are the left-hand HBST images from one person in the two sessions, while Figs. 5.4(c) and

5.4(d) are the right-hand HBST images from the same person. Figs. 5.4(e)~(h) are the left and right-hand HBST images from another person. Fig. 5.5 shows the HBST images from one male subject and one female subject. From these HBST example images, we can have the following observations. 1) First, the left-hand and right-hand HBST patterns of a person are similar. 2) Second, the HBST patterns captured in different sessions from the same person are similar. 3) Third, the HBST patterns from different persons are different, which implies its potential for human identification. 4) At last, the HBST patterns of male and female subjects are different, which makes HBST pattern a good feature for gender classification.

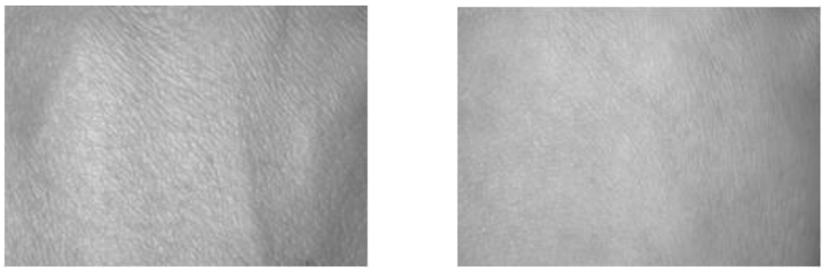


Figure 5.5: (a) and (b) are the HBST images from one male and one female, respectively.

5.3 HBST Feature Extraction and Classification

Texture classification is a classical topic in computer vision and pattern recognition. Although some well-known texture classification methods [17, 20, 47] can obtain good performance on some benchmark databases such as the UIUC [47] and CUReT [89] texture datasets, they may not be suitable for HBST patterns due to the special micro-structure of hand back skin. Since there are no obvious interest points or interest regions in HBST images, the method in [47] cannot detect accurately affine invariant regions for a robust statistical description of the skin texture, and thus it will fail to classify the HBST patterns. The multi-fractal spectrum method [20] and the LBP method in [17] cannot obtain good results either because the feature generated by them cannot characterize the appearance of the skin texture well.

With a more careful look of the HBST images (refer to Fig. 5.4 please), we can observe some properties of the HBST patterns. First, there are no clear edges and corner points in the HBST images. Second, the HBST patterns are often constructed by some micro-cellular structures. Third, those micro-structures are generally distributed uniformly across the whole HBST image. Based on these observations, we choose to learn the micro-structures (i.e., textons) from the training HBST images, and then use them to describe the query HBST image for classification. Our experimental results in Section 5.4 also verify that the texton learning based method performs well for HBST recognition.

By filtering the training images with the MR8 filter bank, for each class of HBST images we can construct a training dataset $\boldsymbol{X} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n]$, where $\boldsymbol{x}_i$, $i = 1,2,...,n$, is an 8-dimensional MR8 filtering response vector at a pixel of the training sample image of this class. A dictionary of textons, denoted by $\boldsymbol{D} = [\boldsymbol{d}_1, \boldsymbol{d}_2, \ldots, \boldsymbol{d}_l]$, can be trained from the constructed training dataset $\boldsymbol{X}$, where $\boldsymbol{d}_j$, $j = 1,2,...,l$, is a texton. The number of textons is generally much smaller than that of the elements in the training dataset, i.e., $l << n$. There are three ways to learn textons from the training dataset $\boldsymbol{X}$: the *K*-means clustering method [50], SR based texton learning method in Chapter 3 and regularized least square based texton learning method in Chapter 4, which are corresponding to different feature extraction methods for classification.

(1). K-means clustering based texton learning and feature extraction with the kd-tree structure

Denote by $\boldsymbol{D}_i$ the texton dictionary for the i^{th} texture class, the dictionary for all c classes of texture images can be formed by amalgamating the c dictionaries, $\boldsymbol{D}=[\boldsymbol{D}_1,\boldsymbol{D}_2,...,\boldsymbol{D}_c]$. With this dictionary $\boldsymbol{D}$, each training texture image can generate a model by mapping it to the texton dictionary.

In Varma and Zisserman's method [50], for each position in a training image, it is labeled with the element in the dictionary $\boldsymbol{D}$ that is closest to the descriptor at this position. Therefore, a histogram can be formed by normalizing the frequencies of texton labels of this image. However, it is time-consuming to find the closest textons in the texton dictionary for all filter response vectors in our task. For example, if 40 textons are learned for each class of skin texture class, the size of the dictionary of all classes of HBST will be 6400. Then, for an HBST image (670×580), it will cost 670×580×6400 linear search operators to encode all pixels in the skin image for feature extraction.

In order to speed up the process of feature extraction, we construct a *kd*-tree structure for the learned textons. The *kd*-tree [114] is a data structure proposed by John Bentley in 1975. Given a set of points in a d-dimensional space, the *kd*-tree is constructed recursively as follows. First, we find a median of the values of the i^{th} coordinate of the points (initially, $i=1$). The i^{th} coordinate of some points are smaller than or equal to the median. The others are greater than the median. Now we can divide the points into two parts by the median. Then the process is repeated recursively with i replaced by $i+1$. The resulting data structure is a binary tree in which every node is a d-dimensional point.

Our goal is to find the closest texton in the texton dictionary for each filter response vector. To this end, we use the nearest neighbor search algorithm [115] to find the texton in the constructed tree which is closest to the given filter response vector. This search can be done efficiently by using the tree properties to quickly eliminate large portions of the search space. For a hand-back skin texture image, the time for encoding all pixels in the skin texture image by finding the closest textons based on the *kd*-tree is about 7 seconds. However, without employing the *kd*-tree structure, it will cost more than 1 minute.

For classification, we use either the nearest neighbor classifier (NNC) with χ^2-distance or the nearest subspace classifier (NSC) for human identification and gender classification.

(2). SR based texton learning and feature extraction

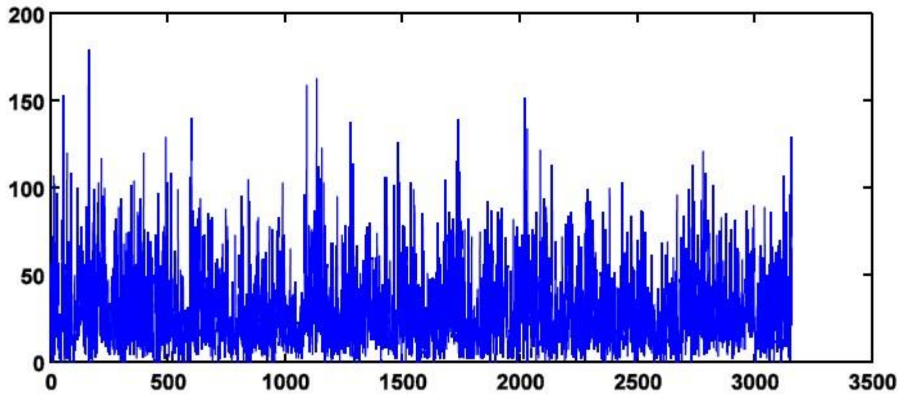
As described in Chapter 3, we can use the technique of sparse representation (SR) to learn an over-complete dictionary of textons via the l_1-norm minimization. And under the SR scheme, we extract the SR coefficient histogram as the HBST feature for recognition. Fig. 5.6 shows some coefficient histograms of HBST images. The NNC and NSC classifiers can be used for classification.

(3). Texton learning by the regularized least square method

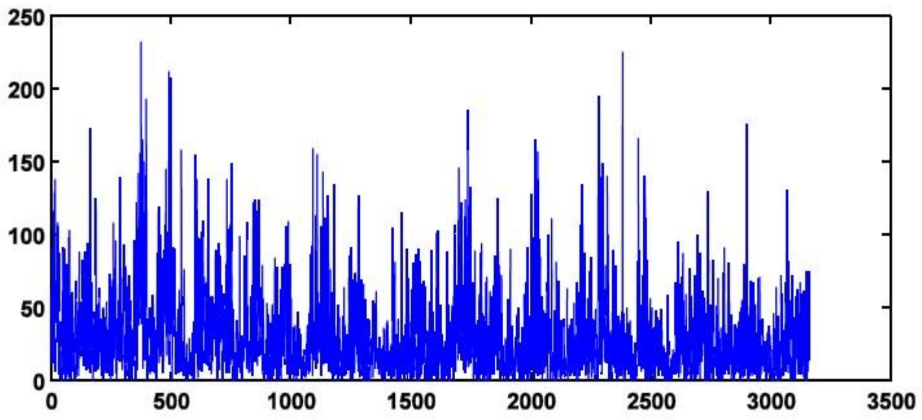
As described in Chapter 4, we learn textons from the training dataset $\boldsymbol{X}$ with the regularized least square, i.e., the objective function (4-1). Then under the regularized least square scheme, a two-stage texton encoding is used to code the training dataset. Finally, two types of statistical features: coding coefficient histogram and coding residual histogram are comined for classificatin. The NNC and NSC classifiers can be used for classification.

(a)

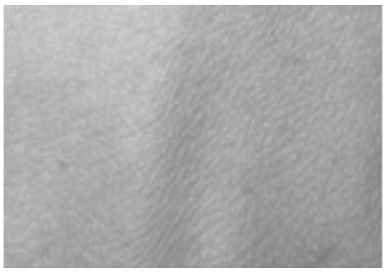
(b)

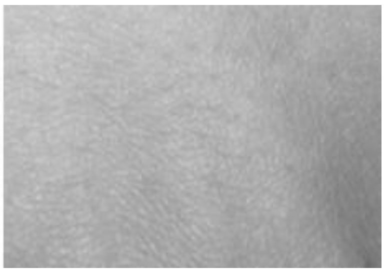


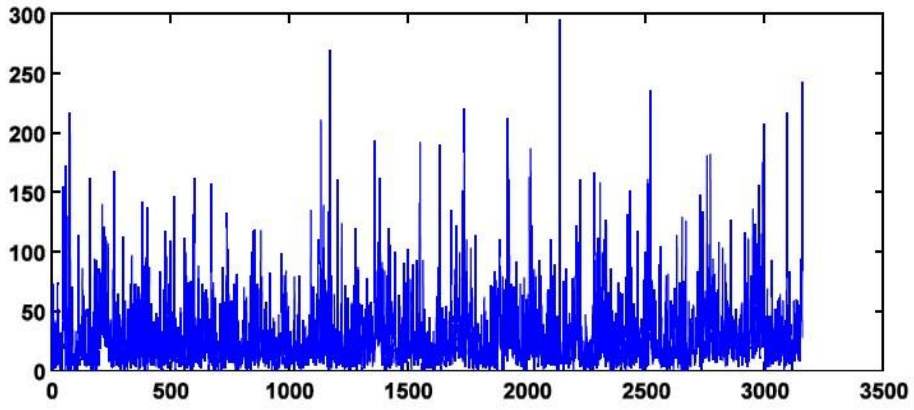
(c)



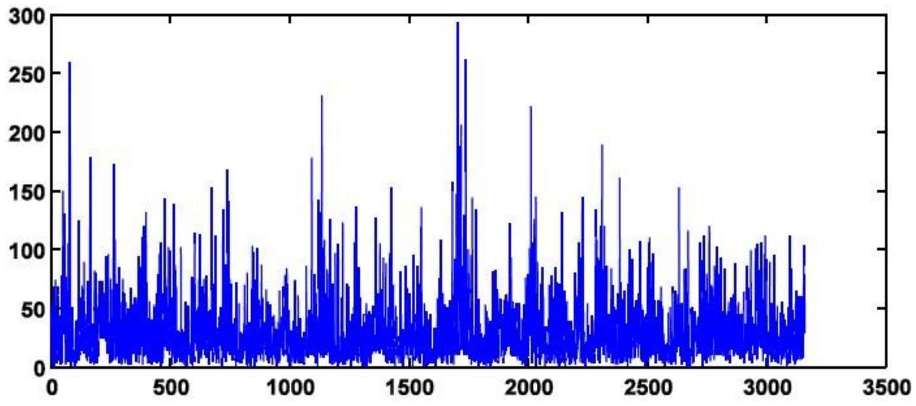
(d)


(e)


(f)



(g)



(h)

Figure 5.6: The coefficient histograms of HBST images from different persons. (c) and (d) are the histograms of the left-hand HBST images (a) and (b) from the same person while (g) and (h) are the histograms of the left-hand HBST images (c) and (d) from another person.

5.4 Experimental Results

5.4.1 Dataset Establishment

In order to evaluate the proposed HBST analysis method for personal identification and gender classification, we established an HBST image dataset using the developed HBST imaging device. Those HBST sample images were collected from 80 volunteers (160 hands), including 61 males and 19 females whose ages are from 20 years old to 50 years old.

The samples were collected in two different sessions. In each session, each person was asked to provide 6 left-hand HBST and 6 right-hand HBST images respectively. Therefore, 12 samples from one person were collected in each session. In total, the database contains 1920 samples from 160 hand backs. The average interval between the first and second sessions is about 30 days, and the maximum and minimum interval is 40 days and 14 days, respectively. In the following experiments, without specific instructions, we use the samples collected in the first session as the training set and the samples in the second session as the test set.

Due to the various difficulties in data collection (e.g., the funding support, the recruitment of volunteers, etc.), our established dataset may not be large and comprehensive enough to make very strong conclusions. Nonetheless, we believe that its size is reasonably large to illustrate if HBST patterns can be used to assist personal identification and gender classification. We are planning to collect more samples from more subjects in the following years, making our dataset more comprehensive and more balanced in terms of male and female subjects.

5.4.2 Personal Identification

In this section we aim to answer the question that whether HBST can be used as a kind of biometric trait to aid personal identification. To this end, we conduct 5 experiments. We denote the texton learning method using the *K*-means clustering [50] in the MR8 feature space and the *kd*-tree structure to label textons, the SR technique in the MR8 feature space and the texton encoding induced statistical feature by VZ_MR8 (*kd*-tree), SR_MR8 and TEISF, respectively. And we compare the proposed method to some representative texture classification methods such as the multi-fractal spectrum method [20], original LBP [17], dominant LBP (DLBP) [19], completed LBP (CLBP) [18]. For the multi-fractal spectrum method, the dimension of the multi-fractal spectrum vector is set to 26. In the original LBP, dominant LBP and completed LBP method, the radius of the neighborhood is 2 and the number of sampled points in the neighborhood is set to 8. For VZ_MR8 (*kd*-tree), 40 textons are learned for each class of HBST images. In SR_MR8, 40 textons are also learned per class. Moreover, in the stage of feature description, for each descriptor, 100 closest textons to $\boldsymbol{x}_i$ in $\boldsymbol{D}$ are chosen to form a sub-dictionary to obtain the SR coefficient. In TEISF, 40 textons are also learned per class and $p = 7$ is set to form a sub-dictionary.

Experiment 1

In the first experiment, all classes of HBST images are involved. The left and right-hand HBST images from the same person are taken as from different classes. Therefore, in this experiment, there are 160 classes and each class has 6 training and 6 testing samples. Since the multi-fractal spectrum vector and the histogram generated by the original LBP method cannot characterize well the appearance (e.g., cell-like micro-structures) of skin texture, they lead to poor experimental results in our task. The multi-fractal spectrum and original LBP methods can only achieve classification accuracies of 35.65% and 46.52%, respectively. Hence, in the following experiments, we only compare VZ_MR8 (*kd*-tree), SR_MR8 and TEISF with DLBP and CLBP.

Table 5.1 shows classification accuracies by the competing methods with NNC and NSC. We can see that the TEISF method by combining the coding coefficient and residual histogram as feature is superior to VZ_MR8 (*kd*-tree) and SR_MR8 for HBST classification. Also, the texton learning based methods (VZ_MR8 (*kd*-tree)， SR_MR8 and TEISF) are better than the CLBP method, which combines the central pixel, magnitude and sign information of the neighborhood to completely model the LBP operator. Moreover, the results with NSC are better than those with NNC.

The interesting HBST image classification accuracies validate that the proposed HBST identification system can well capture the characteristics of skin textures, allowing good discrimination between different classes. These results also suggest that human identification can be aided by HBST analysis as a new biometric trait.

Table 5.1: Classification accuracies by competing methods. For one person, the left-hand and right-hand HBST images are viewed as from two different classes. Thus there are 160 classes in this experiment. (a) shows the results by the NNC classifier, and (b) shows the results by the NSC classifier.

(a) Classification results by NNC

Method	DLBP	CLBP	VZ_MR8(*kd*-tree)	SR_MR8	TEISF
Accuracy	75.56%	84.51%	84.40%	86.81%	88.29%

(b) Classification results by NSC

Method	DLBP	CLBP	VZ_MR8(*kd*-tree)	SR_MR8	TEISF
Accuracy	77.71%	89.14%	88.82%	91.35%	92.51%

Experiment 2

In the second experiment, all HBST images are involved. Different from Experiment 1, here the left and right-hand HBST images from the same person are viewed as from the same class. Therefore, in this experiment there are 80 classes and each class has 12 training and 12 test samples. The experimental results using the DLBP, CLBP, VZ_MR8(*kd*-tree), TL_SR and TEISF method with NNC and NSC are compared in Table 5.2. We can see that for all methods the classification accuracy is increased. This is mainly because the total number of classes is smaller than that in Experiment 1, and the left-hand and right-hand HBST images of one person are similar.

Table 5.2: Classification accuracies by competing methods. For one person, the left-hand and right-hand HBST images are viewed as from the same class. Thus there are 80 classes in this experiment. (a) shows the results by the NNC classifier, and (b) shows the results by the NSC classifier.

(a) Classification results by NNC

Method	DLBP	CLBP	VZ_MR8(*kd*-tree)	SR_MR8	TEISF
Accuracy	78.59%	86.29%	86.40%	90.17%	91.3%

(b) Classification results by NSC

Method	DLBP	CLBP	VZ_MR8(*kd*-tree)	SR_MR8	TEISF
Accuracy	82.34%	90.24%	90.45%	94.09%	95.2%

Experiment 3

The aim of this experiment is to evaluate the performance on the left and right-hand HBST separately. For either left-hand or right-hand HBST images, there are 80 classes and 480 images in the training and test sets, respectively. The classification accuracies by different methods are listed in Tables 5.3 and 5.4. From the experimental results, one can see that the classification accuracy on the right-hand HBST images is slightly higher than that on the left-hand HBST. This is probably because most people who provided their HBST samples to our database are right handed so that they feel more convenient to use our imaging device with the right hand. Therefore, compared to the left-hand HBST samples, the right-hand HBST samples collected in our database have less deformation, which results in a slightly higher classification accuracy for personal identification.

Table 5.3: Classification accuracies on the left-hand HBST images. (a) shows the results by the NNC classifier, and (b) shows the results by the NSC classifier.

(a) Classification results by NNC

Method	DLBP	CLBP	VZ_MR8(*kd*-tree)	SR_MR8	TEISF
Accuracy	80.38%	85.51%	84.54%	88.60%	90.1%

(b) Classification results by NSC

Method	DLBP	CLBP	VZ_MR8(*kd*-tree)	SR_MR8	TEISF
Accuracy	83.43%	89.45%	89.34%	93.67%	94.2%

Table 5.4: Classification accuracies on the right-hand HBST images. (a) shows the results by the NNC classifier, and (b) shows the results by the NSC classifier.

(a) Classification results by NNC

Method	DLBP	CLBP	VZ_MR8(*kd*-tree)	SR_MR8	TEISF
Accuracy	82.91%	86.44%	85.24%	89.71%	90.62%

(b) Classification results by NSC

Method	DLBP	CLBP	VZ_MR8(*kd*-tree)	SR_MR8	TEISF
Accuracy	85.69%	90.46%	89.42%	93.75%	94.82%

Experiment 4

In this experiment, we fuse the left-hand and right-hand HBST for identification. That is, both the left-hand and right-hand HBST samples of a person will be collected to identify his/her identity. Therefore, there are 480 pairs of left-hand samples in the training set, which are from 80 subjects. In the test set there are also 480 pairs of HBST samples. We employ NNC and NSC to fuse the left-hand and right-hand HBST samples for classification. For NNC, we calculate two distances χ_l^2 and χ_r^2, where χ_l^2 is the distance between the left-hand test sample and left-hand training sample, and χ_r^2 is the distance between the right-hand test sample and right-hand training sample from the same pair. Then two distances can be fused together by the simple weighted average method. The final distance for classification is $\chi_f^2 = w \cdot \chi_l^2 + (1-w) \cdot \chi_r^2$, where weight w can be trained from the training dataset using the "leave-one-out" strategy. For the competing classification methods in our experiment, the weights are 0.4, 0.5, 0.45, 0.4 and 0.45, respectively. For NSC, two errors err_l and err_r are computed, where err_l is the error between the left-hand test sample and one class, and err_r is the error between the right-hand test sample and the same class. The final error for classification is $err_f = w \cdot err_l + (1-w) \cdot err_r$. Weight w can also be trained from the training dataset using the "leave-one-out" strategy, and in this experiment, the weights are 0.55, 0.6, 0.65, 0.6 and 0.65, respectively. The classification accuracies by fusing the left-hand and right-hand HBST with different methods with NNC and NSC are listed in Table 5.5. Compared with the results in Experiments 1~3, one can see that the classification accuracy by fusing the left-hand and right-hand HBST images is much increased, showing that the left-hand and right-hand HBST patterns have complementary information.

Table 5.5: Classification accuracies by fusing the left-hand and right-hand HBST. (a) shows the results by the NNC classifier, and (b) shows the results by the NSC classifier.

(a) Classification results by NNC

Method	DLBP	CLBP	VZ_MR8(*kd*-tree)	SR_MR8	TEISF
Accuracy	85.23%	87.24%	89.03%	92.51%	93.2%

(b) Classification results by NSC

Method	DLBP	CLBP	VZ_MR8(*kd*-tree)	SR_MR8	TEISF
Accuracy	89.42%	92.3%	93.75%	96.31%	96.78%

Experiment 5

As we mentioned in the Introduction section, one goal of this work is to investigate whether hand back skin texture patterns can be used to aid other biometrics identifiers to improve personal identification accuracy. Therefore, in this experiment we fuse palmprint and HBST for personal identification. Since there are 160 hand backs (80 left hands and 80 right hands) in our HBST dataset, we randomly extract from the PolyU palmprint database [106] 1920 palmprint images, which belong to 160 palms (80 left hands and 80 right hands). Each palm has 12 samples collected from two separated sessions, 6 samples per session. We then assume that each hand has 6 palmprint images and 6 HBST images in each session, and use the data from the first session for training, while use the data from the second session for testing.

We use the competitive code scheme developed in [125] to extract the palmprint feature and use the Hamming distance to measure the similarity between palmprint features. As in Experiment 4, we fuse palmprint and HBST by the weighted average method with NNC. The final distance for classification is $d_f^2 = w \cdot d_p^2 + (1-w) \cdot d_h^2$, where d_p is the distance between palmprint samples and d_h is the distance between HBST samples. In our experiment, the weight is set to 0.8 by experience. The classification accuracies of palmprint, HBST and the fusion of palmprint and HBST with NNC are listed in Table 5.6. Compared with the identification rate by either palmprint or HBST individually, one can see that the accuracy is much improved by fusing palmprint and HBST matching distances. This validates that HBST can be used to aid the existing biometric traits for person identification.

Table 5.6: Classification accuracies by palmprint, HBST and the fusion of them.

Feature	Palmprint	HBST	Fusion
Accuracy	98.65%	86.81%	99.58%

5.4.3 Gender Classification

As can be seen in Fig. 5.5, the hand back skin appearance differs much from male to female. In most cases, the HBST surface from female is much smoother than that from male, and the size of micro-cells in female HBST samples is smaller than those for males. Therefore, it is very interesting to verify that if the HBST patterns are distinctive enough to distinguish males from females. In this section, we conduct such experiments for gender classification.

In our HBST dataset, there are 61 males and 19 females. We take the samples of both the left-hand and right-hand as samples from the same subject. In gender classification, there are only two classes: male and female. The samples from all 61 male subjects are taken as the samples of the male class, and the samples from all 19 females are taken as those of the female class. The 960 samples collected from the first session are used as the training set, and the other 960 samples from the second session are taken as test samples. Table 5.7 shows the results by the DLBP, CLBP, VZ_MR8 (*kd*-tree), SR_MR8 and TEISF method with NNC and NSC. One can see that the gender classification accuracy can be higher than 98%, which implies that HBST can be aided to distinguish males from females.

Moreover, in Table 5.8 and 5.9, we present the numbers of falsely classified male and female samples by these methods with NNC and NSC. As illustrated in Table 5.8 and 5.9, by the TEISF method with NNC and NSC, the classification error rates of male samples are 1.23% and 0.96%, respectively. The classification error rates of female samples by the TEISF method are 1.32% and 1.32%, respectively. Although the numbers of male and female subjects in our dataset are not balanced, the classification error rate on female samples is only slightly higher than that on male samples with the TEISF method. Certainly, we need to collect more samples and make the dataset more balanced to further validate this conclusion.

Table 5.7: Gender classification accuracies by different methods. (a) shows the results by the NNC classifier, and (b) shows the results by the NSC classifier.

(a) Classification results by NNC

Method	DLBP	CLBP	VZ_MR8(*kd*-tree)	SR_MR8	TEISF
Accuracy	95.42%	97.50%	98.54%	98.65%	98.75%

(b) Classification results by NSC

Method	DLBP	CLBP	VZ_MR8(*kd*-tree)	SR_MR8	TEISF
Accuracy	95.83%	97.71%	98.75%	98.85%	98.96%

Table 5.8: Numbers and rates of falsely classified male samples by different methods. (a) shows the results by the NNC classifier, and (b) shows the results by the NSC classifier.

(a) Numbers and rates of falsely classified male samples by NNC

Method	DLBP	CLBP	VZ_MR8(*kd*-tree)	SR_MR8	TEISF
Number	28	14	9	9	9
Rate	3.83%	1.91%	1.23%	1.23%	1.23%

(b) Numbers and rates of falsely classified male samples by NSC

Method	DLBP	CLBP	VZ_MR8(*kd*-tree)	SR_MR8	TEISF
Number	25	12	8	7	7
Rate	3.42%	1.64%	1.09%	0.96%	0.96%

Table 5.9: Numbers and rates of falsely classified female samples by different methods. (a) shows the results by the NNC classifier, and (b) shows the results by the NSC classifier.

(a) Numbers and rates of falsely classified female samples by NNC

Method	DLBP	CLBP	VZ_MR8(*kd*-tree)	SR_MR8	TEISF
Number	16	10	5	4	3
Rate	7.02%	4.39%	2.19%	1.75%	1.32%

(b) Numbers and rates of falsely classified female samples by NSC

Method	DLBP	CLBP	VZ_MR8(*kd*-tree)	SR_MR8	TEISF
Number	15	10	4	4	3
Rate	6.58%	4.39%	1.75%	1.75%	1.32%

5.4.4 Discussion

In Sections 5.4.2 and 5.4.3, we employed three kinds of dictionary learning methods to learn textons and extract histogram features for personal identification and gender classification. We compare the three methods in terms of speed and accuracy, and list the qualitative comparison results in Table 5.10. From this table, one can see that in terms of the speed, VZ_MR8 (*kd*-tree) is the fastest since the *K*-means clustering is fast and with the *kd*-tree structure labeling the closest texton in the dictionary is also fast. SR_MR8 is slow since solving the l_1 -minimization problem is very time-consuming. Certainly, in terms of accuracy TEISF is the highest due to the fusion of the coefficient histogram and residual histogram.

Table 5.10: Qualitative comparisons of the speed and accuracy of different texton learning methods.

Method	VZ_MR8(*kd*-tree)	SR_MR8	TEISF
Speed	Fastest	Slow	Fast
Accuracy	Medium	High	Highest

Compared with the biometric traits such as fingerprint, palmprint, iris, etc, the personal identification accuracy of HBST is much lower than them. However, each biometric trait has its pros and cons, and no one can supersede the other one. And in practice using two or more biometric traits together will be more robust. In this work, our goal is to investigate whether hand back skin texture patterns can be used to aid personal identification and gender classification. Considering that HBST images can be collected when capturing fingerprint or palmprint images, fusing fingerprint/palmprint and HBST can be a good way for multi-modal biometrics, as described in section 5.4.2. Furthermore, as a specific type of texture images, the established HBST dataset can be used to test texture classification algorithms in the community of computer vision and pattern recognition.

It should be noted that although HBST analysis can assist personal identification and gender classification, there are some factors such as hairs on skin and humidity of skin to affect the performance of personal identification and gender classification. In our established HBST database, most of samples are collected from oriental people so that there are relatively few hairs on the hand back skin. In our future work, we will collect more samples from more subjects and investigate the influences of these factors on skin texture analysis. In addition, modeling skin texture over a long period is a challenging problem since there are large variations between skin textures in different ages. Hence, in the future we will study how to model skin texture over a long period more effectively to improve the performance of biometric tasks with skin texture analysis.

5.5 Summary

This chapter studied the problem of using hand back skin texture (HBST) for personal identification and gender classification. An effective skin texture imaging system was developed for capturing HBST images. Moreover, we employed the texton learning based methods to model the HBST pattern. To evaluate the performance of the proposed system, an HBST dataset was established, consisting of 1920 images from 160 hands of 80 persons. Extensive experiments were conducted and the experimental results showed that human identification and gender classification can be aided by HBST analysis with good performance.

Chapter 6. Conclusions and Future Work

6.1 Conclusions

In this thesis, we first presented a sparse representation based dictionary learning method for texture representation. In traditional texton learning based approaches, textons are usually learned via the *K*-means clustering. However, the *K*-means clustering is too coarse to characterize the complex feature space of textures. Compared to the *K*-means clustering, the sparse representation model can increase the representation accuracy with the linear combination of several textons for dictionary learning. Moreover, the representation coefficient histogram is constructed for texture classification. Experimental results demonstrated that the proposed method can yield better performance than the traditional texture classification method based on the *K*-means clustering.

In order to reduce the computation complexity of the sparse representation (sparse coding) and also achieve a good representation accuracy, we presented an effective and efficient texton encoding scheme for texture classification. First, a regularized least square based texton learning method was developed to learn the dictionary of textons class by class. Second, a fast two-stage l_2-norm texton encoding method was proposed to code the texture feature over the concatenated dictionary of all classes. Third, two types of histogram features were defined and computed from the texton encoding outputs: coding residuals and coding coefficients. Finally, two histogram features were fused for classification via the nearest neighbor classifier and the nearest subspace classifier. Experimental results on three benchmark texture datasets showed that the proposed method is very competitive, achieving probably the best accuracy so far in the literature.

Finally, we studied the hand-back skin texture (HBST) pattern recognition problem with applications to personal identification and gender classification. A specially designed imaging system was developed to capture the HBST images, and an HBST image dataset was established, which consists of 1920 images from 80 persons. Then, three texton learning based texture classification methods were performed on the established dataset: the *K*-means clustering, the sparse representation method and texton encoding induced statistical features with the regularized least square method. The experimental results illustrated that HBST can be used to assist human identification and gender classification. Meanwhile, the established HBST dataset also provides a good platform to evaluate the various texture analysis methods.

6.2 Future Work

Our research in this thesis mainly focused on texton learning and feature extraction for texture classification with an interesting application to skin texture analysis. In the future work, we will improve the proposed methods and investigate more deeply the topic of skin texture analysis.

One issue needs to be further studied is to see if we can reduce the numbers of learned textons and improve the classification accuracy since currently the learned atoms from different classes are correlated. If the textons for different classes can be learned jointly, a more compact yet more effective dictionary can be expected;

nonetheless, the joint learning of such a texton dictionary is much more complicated than the texton learning class by class and this will be investigated in our future research. In addition, in the texton learning methods proposed in this thesis, the scale and viewpoint change of texture images is not explicitly considered. Therefore, when there are large scale and viewpoint changes, how to learn efficiently and effectively the textons which are robust to these changes should be well studied in the future.

For HBST analysis, currently, modeling skin texture over a long period is still a challenging problem since there are large variations between skin textures in different ages. Hence, in the future we will focus on how to model skin texture over a long period to improve the performance of some biometric tasks with skin texture analysis. Meanwhile, we will investigate influences of some factors such as hairs and humidity on skin texture analysis. In addition, more HBST samples need to be collected to verify the different aspects of HBST analysis and the algorithm development.

Bibliography

[1] C. Schmid, “Constructing models for content –based image retrieval”, in *CVPR*, pp. 39–45, 2001.

[2] J. Zujovic, T. N. Papps, and D. L. Neuhoff, “Perceptual similarity metrics for retrieval of natural textures”, in *ICIP*, pp. 2225–2228, 2009.

[3] S. C. Zhu, Y. N. Wu, and D. B. Mumford, “Filters, random fileds and maximum-entropy (FRAME): Towards a unified theory for texture modeling”, *IJCV*, vol. 27, no. 2, pp. 107-126, 1998.

[4] F. G. Meyer, A. Z. Averbuch, and J. O. Stromberg, “Fast adaptive wavelet packet image compression”, *IEEE Trans. IP*, vol. 9, no. 5, pp. 792-800, 2000.

[5] A. Efros, and T. Leung, “Texture synthesis by non-parametric sampling”, in *ICCV*, vol. 2, pp. 1039-1046, 1999.

[6] T. Lindeberg, and J. Garding, “Shape from texture from a multi-scale perspective”, in *ICCV*, pp. 683-691, 1993.

[7] J. Malik, and R. Rosenholtz, “Computing local surface orientation and shape from texture for curved surfaces”, *IJCV*, vol. 23, no. 2, pp. 149-168, 1997.

[8] D. James, B. D. Clymer, and P.Schmalbrock, “Texture detection of simulated microcalcification susceptibility effects in magnetic resonance imaging of breasts”, *Journal of Magnetic Resonance Imaging*, vol. 13, no. 6, pp. 876-881, 2001.

[9] P. Miller, and S. Astley, “Classification of breast tissue by texture analysis”, *Image and Vision Computing*, vol. 10, no. 5, pp. 277-282, 1992.

[10] N. Petrick, H. P. Chan, D. Wei, “Automated detection of breast masses on mammograms using adaptive contrast enhancement and texture classification”, *Medical Physics*, vol. 23, no. 10, pp. 876-881, 1996.

[11] O. G. Kula, K. J. Dana, F. P. Murphy, and B. K. Rao, “Bidirectional imaging and modeling of skin texture”, *IEEE Trans. Biomedical Engineering*, vol. 23, no. 10, pp. 876-881, 1996.

[12] J. Shotton, J. Winn, C. Rother, A. Criminisi, “TextonBoost for image understanding: multi-class object recognition and segmentation by jointly modeling texture, layout and context”, *IJCV*, vol. 81, no. 1, pp. 876-881, 2009.

[13] J. Shotton, A. Blake, and R. Cipolla, “Efficiently combining contour and texute cues for object recognition”, in *BMVC*, 2008.

[14] A. Lorette, X. Descombes, and J. Zerubia, “Texture analysis through a markovian modeling and fuzzy classification”, *IJCV*, vol. 36, no. 3, pp. 221-236, 2000.

[15] L. A. Ruiz, A. Fdez-sarria, and J. A. Recio, “Texture feature extraction for classification of remote sensing data using wavelet decomposition: a comparative study”, in *ISPRS*, pp. 1109-1114, 2004.

[16] R.M.Haralick, K.Shanmugam, and I.Dinstein, “Textural features for image classification”, *IEEE Trans. SMC*, vol. 3, no. 6, pp. 610-621, 1973.

[17] T.Ojala, M.Pietikainen, and T.Maenpaa, “Multi-resolution gray-scale and rotation invariant texture classification with local binary patterns”, *IEEE Trans. PAMI*, vol. 24, no. 7, pp. 971–987, 2004.

[18] Z. H. Guo, L. Zhang, and D. Zhang, “A Completed Modeling of Local Binary Pattern Operator for Texture Classification”, *IEEE Trans. IP*, vol. 19, no. 6, pp. 1657-1663, 2010.

[19] S. Liao, W. K. Law, and C. S. Chung, “Dominant local binary patterns for texture classification”, *IEEE Trans. IP*, vol. 18, no. 3, pp. 1107-1118, 2009.

[20] Y. Xu, H. Ji, and C. Fermuller, “Viewpoint invariant texture description using fractal analysis”, *IJCV*, vol. 83, no. 1, pp. 85-100, 2009.

[21] Y. Xu, X. Yang, H. Ling, and H. Ji, “A new texture descriptor using multifractal analysis in multi-orientation wavelet pyradmid”, in *CVPR*, 2010.

[22] Y. Xu, H. Ji, and C. Fermuller, “A projective invariant for textures”, in *CVPR*, pp. 1932-1939, 2006.

[23] S. Qian, and D. P. Chen, “Discrete Gabor transform”, *IEEE Trans. SP*, vol. 41, no. 7, pp. 2429-2438, 1993.

[24] S. Mallat, “A theory for multiresolution signal decomposition: the wavelet transform”, *IEEE Trans. PAMI*, vol. 11, no. 1, pp. 674-693, 1993.

[25] E. P. Simoncelli, and W. T. Freeman, “The steerable pyramid: a flexible architecture for multi-scale derivative computation”, in *ICIP*, 1995.

[26] J. Portilla, and E. P. Simoncelli, “A parametric texture model based on joint statistics of complex wavelet coefficients”, *IJCV*, vol. 40, no. 1, pp. 49-71, 2001.

[27] M. N. Do, and M. Vetterli, “The contourlet transform: an efficient directional multiresolution image representation”, *IEEE Trans. IP*, vol. 14, no. 12, pp. 2091-2106, 2005.

[28] K. O. Cheng, N. F. Law, and W. C. Siu, “Multiscale directional filter bank with applications to structured and random texture retrieval”, *Pattern Recognition*, vol. 40, no. 5, pp. 1182-1194, 2006.

[29] C. M. Pun, M.C. Lee, “Log-polar wavelet energy signatures for rotation and scale invariant texture classification”, *IEEE Trans. PAMI*, vol. 26, no. 9, pp. 1228–1333, 2004.

[30] K. J. Khouzani, and H. S. Zadeh, “Rotation invariant multiresolution texture anylysis using radon and wavelet transforms”, *IEEE Trans. IP*, vol. 14, no. 6, pp. 783–794, 2005.

[31] K. J. Khouzani, and H. S. Zadeh, “Radon transform orientation estimation for rotation invariant texture analysis”, *IEEE Trans. PAMI*, vol. 27, no. 6, pp. 1004–1008, 2005.

[32] R. Porter, and N. Canagarajah, “Robust rotation invariant texture classification: wavelet, Gabor filter and GMRF based schemes”, in *Vision, Image and Signal Processing*, vol. 144, no. 3, pp. 180–188, 1997.

[33] J. L. Chen, and A. Kundu, “Rotation and grayscale transform invariant texture identification using wavelet decomposition and HMM”, *IEEE Trans. PAMI*, vol. 16, no. 2, 1994.

[34] W. R. Wu, and S. C. Wei, “Rotation and gray scale transform invariant texture classification using spiral resampling, subband decomposition and hidden Markov model”, *IEEE Trans. IP*, vol. 5, no. 10, pp. 1423–1434, 1996.

[35] M. Do, M. vetterli, “Wavelet-based texture retrieval using generalized Gaussian density and Kullback-Leibler distance”, *IEEE Trans. IP*, vol. 14, no. 10, pp. 146–158, 2002.

[36] S. D. Kim, and S. Udpa, “Texture classification using rotated wavelet filter”, *IEEE Trans. SMC*, vol. 30, no. 6, pp. 847–852, 2000.

[37] M. Kokare, P. K. Biswas, and B. N. Chatterji, “Rotation invariant texture image retrieval using rotated complex wavelet filters”, *IEEE Trans. SMC*, vol. 36, no. 6, pp. 1273–1282, 2006.

[38] G. M. Haley, and B. S. Manjunath, “Rotation invariant texture classification using a complete space-frequency model”, *IEEE Trans. IP*, vol. 8, no. 2, pp. 255–269, 1999.

[39] H. Greenspan, S. Belongie, R.Goodman, and P.Perona, “Rotation invariant texture recognition using a steerable pyramid”, in *ICPR*, pp. 162–167, 1994.

[40] B. Julesz, “Textons, the elements of texture perception, and their interactions”, *Nature*, vol. 290, pp.91–97, 1981.

[41] T. Leung, and J. Malik, “Representing and recognizing the visual appearance of materials using three-dimensional textons”, *IJCV*, vol. 43, no. 2, pp. 29–44, 2001.

[42] K. J. Dana, and S. K. Nayar, “Histogram model for 3D textures”, in *CVPR*, pp. 618–624, 1998.

[43] O. G. Cula, and K. J. Dana, “Compact representation of bidirectional texture functions”, in *CVPR*, pp. 1041–1047, 2001.

[44] O. G. Cula, and K. J. Dana, “3D texture recognition using bidirectional feature histograms”, *IJCV*, vol. 59, no. 1, pp. 33–60, 2004.

[45] K. Mikolajczyk, and C. Schmid, “Scale and affine invariant interest point detector”, *IJCV*, vol. 60, no. 1, pp. 63–86, 2004.

[46] D. Lowe, “Distinctive image features from scale-invariant features”, *IJCV*, vol. 60, no. 2, pp. 91–110, 2004.

[47] S. Lazebnik, C. Schmid, and J. Ponce, “A sparse texture representation using local affine regions”, *IEEE Trans. PAMI*, vol. 27, no. 2, pp. 1265–1278, 2005.

[48] J. Zhang, M. Marszalek, S. Lazebnik, and C. Schmid, “Local features and kernels for classification of texture and object categories: a comprehensive study”, *IJCV*, vol. 73, no. 2, pp. 213–238, 2007.

[49] M. Varma, and A. Zisserman, “Classifying images of materials: achieving viewpoint and illumination independence”, in *ECCV*, pp. 255–271, 2002.

[50] M. Varma, and A. Zisserman, “A statistical approach to texture classification from single images”, *IJCV*, vol. 62, no. 2, pp. 61–81, 2005.

[51] M. Varma, and A. Zisserman, “Texture classification: are filter banks necessary?”, in *CVPR*, pp. 691–698, 2003.

[52] M. Varma, and A. Zisserman, “A statistical approach to material classification using patch exemplars”, *IEEE Trans. PAMI*, vol. 27, no. 2, pp. 2032–2047, 2009.

[53] M. Varma, and A. Zisserman, “Loally invariant fractal features for statistical texture classification”, in *ICCV*, 2007.

[54] L.Liu, and P.Fieguth, “Texture classification from random features”, *IEEE Trans. PAMI*, 2011.

[55] L. Liu, P. Fieguth, G. Kuang, and H. Zha, “Sorted random projections for robust texture classification”, in *ICCV*, 2011.

[56] L. Liu, P. Fieguth, and G. Kuang, “Combined sorted random features for texture classification”, in *ICIP*, 2011.

[57] E. Candes, and T. Tao, “Near-optimal signal recovery from random projections: universal encoding strategies?”, *IEEE Trans. Information Theory*, vol. 52, no. 12, pp. 5406–5425, 2006.

[58] D. Donoho, “Compressive sensing”, *IEEE Trans. Information Theory*, vol. 52, no. 4, pp. 1289–1306, 2006.

[59] E. Levina, P. Bickel, “The Earth Mover’s distance is the mallows distance: some insights from statistics”, in *ICCV*, pp. 251–256, 2001.

[60] R. Tibshirani，“Regression shrinkage and selection via the lasso”, *SIAM. Imaging Sciences*, vol. 58, no. 4, pp. 267–288, 1996.

[61] S. Chen, D. Donoho, and M. Saunders, “Atomic decomposition by basis pursuit”, *SIAM. Review*, vol. 43, no. 2, pp. 129–159, 2001.

[62] D. Donoho, M. Elad, “Optimal sparse representation in general (nonorthogonal) dictionaries via l_1 minimization”, in *Proceedings of the Natinal Academy of Sciences*, 2003.

[63] A. Bruckstein, D. Donoho, and M. Elad, “From sparse solutions of systems of equations to sparse modeling of signals and images”, *SIAM. Review*, vol. 51, no. 3, pp. 34–81, 2009.

[64] M. Aharon, M. Elad, and A. Bruckstein, “The K-SVD: an algorithm for designing of overcomplete dictionaries for sparse representation”, *IEEE Trans. Signal Processing*, vol. 54, no. 12, pp. 4311–4322, 2006.

[65] J. Martial, M. Elad, and G. Sapiro, “Sparse representation for color image restoration”, *IEEE Trans. IP*, vol. 17, no. 1, pp. 53–69, 2008.

[66] J. Martial, G. Sapiro, and M. Elad, “Learning multiscale sparse representations for image and video restoration”, *SIAM. Multiscale Modeling and Simulation*, vol. 7, no. 1, pp. 214–241, 2008.

[67] R. Rubinstein, M. Zibulevsky, and M. Elad, “Double sparsity: learning sparse dictionaries for sparse signal approximation”, *IEEE Trans. SP*, vol. 58, no. 3, pp. 1553–1564, 2010.

[68] J. Martial, F. Bach, J. Ponce, G. Sapiro, and A. Zisserman, “Non-local sparse models for image restoration”, in *ICCV*, pp. 2272–2279, 2009.

[69] J. Yang, K. Yu, Y. Gong, and T. Huang, “Linear spatial pyramid matching using sparse coding for image classification”, in *CVPR*, 2009.

[70] J. Yang, K. Yu, and T. Huang, “Supervised translation-invariant sparse coding”, in *CVPR*, 2010.

[71] J. Yang, K. Yu, and T. Huang, “Efficient highly over-complete sparse coding using a mixture model”, in *ECCV*, 2010.

[72] J. Wright, A. Yang, A. Ganesh, S. Sastry, and Y. Ma, “Robust face recognition via sparse representation”, *IEEE Trans. PAMI*, vol. 31, no. 2, pp. 210–227, 2009.

[73] E. Lepennec, and S. Mallat, “Sparse geometric image representation with bandelets”, *IEEE Trans. IP*, vol. 14, no. 4, pp. 423–438, 2005.

[74] E. Lepennec, and S. Mallat, “Discrete bandelets with geometric orthogonal filters”, in *ICIP*, 2005.

[75] M. Do, and M. Vetterli, “The finite ridgelet transform for image representation”, *IEEE Trans. IP*, vol. 12, no. 4, pp. 16–28, 2003.

[76] E. Candes, and D. Donoho, “Continuous curvelet transform: I. Resolution of the wavefront set”, *Applied and Computational Harmonic Analysis*, vol. 19, no. 5, pp. 162–197, 2003.

[77] E. Candes, and D. Donoho, “Continuous curvelet transform: II. Discretization and frames”, *Applied and Computational Harmonic Analysis*, vol. 19, no. 5, pp. 198–222, 2003.

[78] S. Mallat, and S. Zhang, “Matching pursuits with time-frequency dictionaries”, *IEEE Trans.SP*, vol. 41, no. 12, pp. 3397–3415, 1993.

[79] J. Tropp, “Greed is good: algorithmic results for sparse approximation”, *IEEE Trans.*

Information Theory, vol. 50, no. 10, pp. 2231–2242, 2004.

[80] E. Bradley, H. Trevor, J. Iain, and T. Robert, “Least angle regression”, *Annals of statistics*, vol. 32, no. 2, pp. 407–499, 2004.

[81] M. Figueiredo, R. Nowak, and S. Wright, “Gradient projection for sparse reconstruction: application to compressed sensing and other inverse problems”, *IEEE Trans. Selected Topics in Signal Processing*, vol. 1, no. 4, pp. 586–597, 2007.

[82] S. Kim, K. Koh, M. Lustig, S. Boyd, and D. Gorinevsky, “An interior-point method for large scale l_1-regularized least squares ”, *IEEE Trans. Selected Topics in Signal Processing*, vol. 1, no. 4, pp. 606–617, 2007.

[83] D. Malioutov, M. Cetin, and A. Willlsky, “Homotopy continuation for sparse signal representation”, In *ICASSP*, 2005.

[84] M. Asif and J. Romberg, “Dynamic updating for l_1 minimization”, *IEEE Trans. Selected Topics in Signal Processing*, vol. 4, no. 2, pp. 421–434, 2010.

[85] A. Beck, and M. Teboulle, “A fast iterative shrinkage-thresholding algorithm for linear inverse problems”, *SIAM. Imaging Science*, vol. 2, no. 1, pp. 183–202, 2009.

[86] Y. Nesterov, “A method of solving a convex programming problem with convergence rate”, *Soviet Mathematics Doklady*, vol. 27, no. 2, pp. 372–376, 1983.

[87] J. Yang, and Y. Zhang, “Alternating direction algorithms for l_1-problems in compressive sensing”, *SIAM. Scientific Computing*, vol. 27, no. 2, pp. 250–278, 2011.

[88] M. Crosier, L. Griffin, “Using basic image features for texture classification”, *IJCV*, vol. 88, no. 2, pp. 447–460, 2010.

[89] K. Dana, B. VanGinneken, S. Nayar, and J. Koenderink, “Reflectance and texture of real world surfaces”, *ACM. Graphics*, vol.18, no. 2, pp. 1–34, 1999.

[90] E. Hayman, B. Caputo, M. Fritz, and O. Eklundh, “On the significance of real-world conditions for material classification”, in *ECCV*, 2004.

[91] http://www.dermnet.com.

[92] O. Cula, K. Dana, F. Murphy, and B. Rao, “Bidirectional imaging and modeling of skin texture”, *IEEE Trans. Biomedical Engineering* , vol.51, no.12, pp. 2148-2159, 2004.

[93] O. Cula, K. Dana, F. Murphy, and B. Rao, “Skin texture modeling”, *IJCV*, vol.62, no.1, pp. 2148-2159, 2005.

[94] H. Tanaka, G. Nakagami, H. Sanada, “Quantitative evaluation of elderly skin based on digital image analysis”, *Skin research and technology*, vol.14, no.2, pp. 192-200, 2008.

[95] K. Kim, Y. Choi, E. Hwang, “Wrinkle feature-based skin age estimation scheme”, in: *Proceedings of the International Conference on Multimedia & Expro*, pp.1222-1225, 2009.

[96] A.K. Jain, P. Flynn, A. Ross, *Handbook of biometrics*, Springer, 2007.

[97] D. Maltoni, D. Maio, A.K. Jain, S. Prabhakar, *Handbook of fingerprint recognition*, Springer, 2003.

[98] N. Ratha, R. Bolle, *Automatic fingerprint recognition systems*, Springer, 2004.

[99] K. Delac, M. Grgic, *Face recognition*, I-Tech Education and Publishing, 2007.

[100] H. Wechsler, *Reliable face recognition methods-system design, implementation and evaluation*, Springer, 2006.

[101] J. Daugman, "High confidence visual recognition of persons by a test of statistical independence", *IEEE Trans. PAMI*, vol.15, no.11, pp. 1148-1161, 1993.

[102] J. Daugman, "How iris recognition works", *IEEE Trans. Circuits and Systems for Video Technology*, vol.14, no.1, pp. 21-30, 2004.

[103] R. B. Hill, *Retinal identification*, in Biometrics: Personal Identification in Networked Society, Kluwer Academic, 1999.

[104] H. Borgen, P. Bours, S.D. Wolthusen, "Visible-spectrum biometric retina recognition", in: *Proceedings of the International Conference on Intelligent Information Hiding and Multimedia Signal Processing*, pp.1056-1062, 2008.

[105] Z. H. Guo, D. Zhang, L. Zhang, W.M. Zuo, "Palmprint verification using binary orientation co-occurrence vector", *Pattern Recognition Letters*, vol.30, no.13, pp. 1219-1227, 2009.

[106] D. Zhang, W. K. Kong, J. You, M. Wong, "online palmprint identification", *IEEE Trans. PAMI*, 2001.

[107] A. Kong, D. Zhang, M. Kamel, "Palmprint identification using feature-level fusion", *Pattern Recognition*, vol.39, no.3, pp. 478-487, 2006.

[108] Z. N. Sun, T. N. Tan, Y.H. Wang, S.Z. Li, "Ordinal palmprint representation for personal identification", in *CVPR*, pp. 279-284, 2005.

[109] L. Zhang, L. Zhang, D. Zhang and H.L. Zhu, "Online finger-knuckle-print verification for personal authentication", *Pattern Recognition*, vol.43, no.7, pp. 2560-2571, 2010.

[110] R. Rowe, "Biometrics based on multispectral skin texture", in *ICB*, pp.1144-1153, 2007.

[111] S. Chen, D. Donoho, and M. Saunders, "Atomic decomposition by basis pursuit", *SIAM. Review*, vol.43, no.1, pp. 129-159, 2001.

[112] H. Lee, A. Battle, R. Raina, and A. Ng, "Efficient sparse coding algorithms", in *NIPS*, pp. 801-808, 2006.

[113] J. Xie, L. Zhang, J. You, and D. Zhang, "Texture classification via patch-based sparse texton learning", in *ICIP*, pp. 2737-2740, 2010.

[114] J. Bentley, "Multidimensional binary search trees used for associative searching",

Communications of the ACM, vol.18, no.1, pp. 509-517, 1975.

[115] J. Bentley, J. Friedman, and R. Finkel, “An algorithm for finding best matches in logarithmic expected time”, *ACM Trans. Mathematical Software*, vol.3, no.1, pp. 209-226, 1977.

[116] V. Bruce, A. Burton, N. Dench, E. Hanna, P. Healey, O. Mason, A. Coombes, R. Fright, and A. Linney, “Sex discrimination: how do we tell the difference between male and female faces?”, *Perception*, vol.22, pp. 131-152, 1993.

[117] B. Moghaddam, and M. Yang, “Learning gender with support faces”, *IEEE Trans. PAMI*, vol.24, pp. 707-711, 2002.

[118] X. Li, S. Maybank, S. Yan, D. Tao, and D. Xu, “Gait components and their application to gender classification”, *IEEE Trans. SMC*, vol.38, pp. 145-155, 2008.

[119] B. Julesz, E. Gilbert, L. Shepp, and H. Frisch, “Inability of humans to discriminate between visual textures that agree in second-order statistics-revised”, *Perception*, vol.2, pp. 391-405, 1973.

[120] R. Broadhurst, “Statistical estimation of histogram variation for texture classification”, in *proceedings of the fourth international workshop on texture analysis and synthesis*, pp. 1597-1604, 2005.

[121] T. Caelli, and B. Julesz, “On perceptual analyzers underlying visual texture discrimination”, *Biological Cybernetics*, vol.25, pp. 167-175, 1978.

[122] M. Mellor, B. Hong, and M. Brady, “Locally rotation, contrast, and scale invariant descriptors for texture analysis”, *IEEE Trans. PAMI*, vol.30, pp. 52-61, 2008.

[123] S. C. Zhu, C.E. Guo, Y.Z. Wang, and Z.J. Xu, “What are textons?”, *IJCV*, vol.62, pp. 145-155, 2005.

[124] S. Tatsumi, H. Noda, and S. Sugiyama, “Estimation of age by epidermal image processing”, *Legal Medicine*, vol.1, pp. 226-230, 1999.

[125] A. Kong, D. Zhang, “Competitive coding scheme for palmprint verification”, In: *International Conference on Pattern Recognition*, 2004.