Feature Coding in Image Classification: A Comprehensive Study

Yongzhen Huang, *Member*, *IEEE*, Zifeng Wu, Liang Wang, *Senior Member*, *IEEE*, and Tieniu Tan, *Fellow*, *IEEE*

Abstract—Image classification is a hot topic in computer vision and pattern recognition. Feature coding, as a key component of image classification, has been widely studied over the past several years, and a number of coding algorithms have been proposed. However, there is no comprehensive study concerning the connections between different coding methods, especially how they have evolved. In this paper, we first make a survey on various feature coding methods, including their motivations and mathematical representations, and then exploit their relations, based on which a taxonomy is proposed to reveal their evolution. Further, we summarize the main characteristics of current algorithms, each of which is shared by several coding strategies. Finally, we choose several representatives from different kinds of coding approaches and empirically evaluate them with respect to the size of the codebook and the number of training samples on several widely used databases (15-Scenes, Caltech-256, PASCAL VOC07, and SUN397). Experimental findings firmly justify our theoretical analysis, which is expected to benefit both practical applications and future research.

Index Terms—Image classification, feature coding, bag-of-features

1 Introduction

1.1 Motivation

Image classification is to assign one or more category labels to an image. It is one of the most fundamental problems in computer vision and pattern recognition, and has a wide range of applications, for example, video surveillance [1], image and video retrieval [2], web content analysis [3], human-computer interaction [4], and biometrics [5]. The bag-of-features (BoF) [6], developed from the bag-of-words model in document analysis [7], is probably the most popular and effective image classification framework in the recent literature. It has achieved the state-of-the-art performance in several databases (e.g., 15-Scenes [8] and Caltech-256 [9]) and competitions (e.g., PASCAL VOC [10] and ImageNet [11]).

Generally, there are five basic steps in the BoF framework used for image classification, as shown in Fig. 1. These steps are, respectively:

1. *Extract patches*. With the images as the input, the outputs of this step are image patches. This process is implemented via sampling local areas of images, usually in a dense (e.g., using fixed grids [12]) or sparse (e.g., using feature extractors [13], [14], [15]) manner.
2. *Represent patches*. Given image patches, the outputs of this step are their feature descriptors (vectors). This process is usually implemented via statistical analysis over pixels of image patches. For example, the popular scale-invariant feature transform (SIFT) descriptor [16] describes a patch with the local accumulation of the magnitude of pixel gradients in each orientation, and finally generates a histogram vector with 128 dimensions (16 subregions multiplied by eight orientations). Other widely used descriptors include local binary pattern [17], histogram of oriented gradients [18], and so on. Extensive studies about feature descriptors can be found in [19], [20].
3. *Generate codewords*. The inputs of this step are feature descriptors extracted from all training images and the outputs are codewords. For computational efficiency, in real application, usually a subset of descriptors is randomly sampled from all descriptors as the input. The codewords are typically generated by clustering (e.g., K-means [21]) over feature descriptors or codeword learning in a supervised [22], [23], [24] or an unsupervised [25], [26], [27] manner. All codewords compose a codebook.
4. *Encode features*. Given feature descriptors and codewords as the input, the output of this step is a coding matrix. In this step, each feature descriptor activates a number of codewords, and generates a coding vector, whose length is equal to the number of codewords. The difference of various coding algorithms lies in how to activate the codewords, i.e., which codewords are activated and how large the amplitudes of their responses are. All coding vectors form a coding matrix.
5. *Pool features*. The input of this step is a coding matrix and the output is a pooling vector for each image, namely the final representation of an image. This step

• *The authors are with the Center for Research on Intelligent Perception and Computing (CRIPAC), National Laboratory of Pattern Recognition (NLPR), Institute of Automation, Chinese Academy of Sciences (CASIA), No. 95 ZhongGuanCun East Street, Beijing 100190, China. E-mail: {yzhuang, zfwu, wangliang, tnt}@nlpr.ia.ac.cn.*

Manuscript received 25 Aug. 2012; revised 7 Apr. 2013; accepted 5 June 2013; published online 13 June 2013.
Recommended for acceptance by A. Torralba.
For information on obtaining reprints of this article, please send e-mail to: tpami@computer.org, and reference IEEECS Log Number TPAMI-2012-08-0660.
Digital Object Identifier no. 10.1109/TPAMI.2013.113.

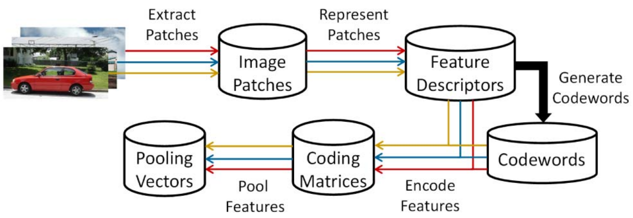

Fig. 1. The general pipeline of the BoF framework for image classification.

is implemented via integrating all responses on each codeword into one value. Classic pooling methods include average pooling (i.e., preserving the average response [6]) and MAX pooling, (i.e., preserving the maximum response [25]). An in-depth analysis on feature pooling can be found in [28], [29].

Of all the above five steps, feature coding is the core component, which links feature extraction and feature pooling, and greatly influences image classification in terms of both accuracy and speed. Although many methods have been presented to promote the development of feature coding, there is still no work that comprehensively studies this exciting field. Our work in this paper makes such a timely survey, in which various coding methods are introduced, their relations are exploited, and existing problems and open directions are discussed. We believe that this work will greatly benefit both beginners and practitioners in the field.

1.2 Taxonomy

For clarity, we group the existing coding strategies into five major categories according to their motivations, as shown in the right part of Fig. 2.

Global coding is generally designed to estimate the probability density distribution (PDD) of features. It focuses on the global description of all features rather than each individual feature. There are mainly two kinds of strategies in global coding:

- *Voting-based methods* [6], [30] describe the distribution of features with a histogram, which carries the occurrence information of codewords. Such a histogram is usually constructed by hard quantization or soft quantization.
- *Fisher coding-based methods* [31], [32] estimate the distribution of features with the Gaussian mixture models (GMM), consisting of the weights, the means, and the covariance matrix of multiple Gaussian distributions, each of which reflects one pattern of features.

Local coding is proposed to describe each individual feature. Three kinds of local coding methods have been developed:

- *Reconstruction-based methods* [25], [26], [33] use a small part of codewords to describe each feature via solving a least-square-based optimization problem with constraints on codewords.
- *Local tangent-based methods* [34], [35] derive an exact description for each feature through approximating the Lipschitz smooth manifold where features are located.

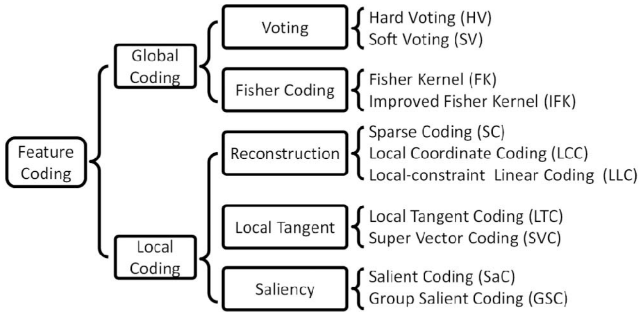

Fig. 2. A taxonomy of coding methods in image classification. Several representatives are listed for each type of coding scheme.

- *Saliency-based methods* [36], [37] encode each feature by the saliency degree, which is calculated using the ratio or the difference of the distances from a feature to its nearby codewords.

It should be noted that the use of the concepts "global" and "local" is to keep consistent with the motivations presented in the original papers. Discussion on the above five kinds of feature coding methods will be detailed in Section 2.

1.3 Difference from Previous Work

Our work is most related to Chatfiled et al. [38]. However, there are several important differences:

1. The major concerns of two works are different. Their work focuses on implementation details for *experimental comparison*, while our work emphasizes *theoretical study*, i.e., the motivations of coding methods as well as their underlying relations. The experimental part in our work is mainly for justifying the theoretical analysis.
2. The criterions of categorizing coding methods are different. Their work groups coding methods into two categories according to their *final representations*: 1) expressing features as the combination of codewords and 2) recording the difference between features and codewords. We divide coding methods into five typical categories based on their different *principles*, which is more helpful for theoretical analysis. For example, although improved Fisher kernel (IFK) and super-vector coding (SVC) are similar in the final representation, IFK estimates global probability density distribution, and SVC, derived from LCC, pursues exact description of each feature.
3. The focuses of experimental evaluation are different. Their work evaluates more on implementation details, for example, sampling density, normalization, flipping images, and kernels, mainly on the VOC07 database, while our work evaluates more on coding algorithms, codebook sizes, numbers of training samples, and databases (including the more challenging Caltech256 and SUN397 databases).

1.4 Contributions

The major contributions of this paper are summarized as follows:

- Provide a survey on recent progress in feature coding, including the motivations and mathematical representations of different categories of coding methods. This part will be especially beneficial for the beginners to get familiar with this research field.

- Exploit the relations among different kinds of coding methods, based on which we propose for the first time a taxonomy and the corresponding evolution map. This part provides an in-depth understanding to this area.
- Evaluate several representative coding algorithms. Some meaningful findings are obtained, which is useful for practical applications.
- Summarize main problems and challenges of current studies, and point out some open directions in future.

The rest of this paper is organized as follows: Section 2 provides a survey on various coding methods. Section 3 exploits the relations among different coding methods. Section 4 empirically evaluates representative coding algorithms on four databases. Finally, Section 5 concludes the paper with discussions on future research.

2 Coding Methods

In this section, different kinds of coding methods are discussed according to the proposed taxonomy depicted in Fig. 2. Let $\mathcal{X} = [x_1, x_2, \ldots, x_N] \in \mathbb{R}^{D\times N}$ be N D-dimensional features extracted from an image, $\mathcal{B} = [b_1, b_2, \ldots, b_M] \in \mathbb{R}^{D\times M}$ be a codebook with M codewords (typically obtained by clustering over features), and $\mathcal{V} = [v_1, v_2, \ldots, v_N]$ be the corresponding representation of these N features. In feature coding, each x is represented by the codebook $\mathcal{B}$. This process will generate responses on M codewords, consisting of a coding vector v with M elements. For most coding algorithms, only a part of codewords will be chosen to represent a feature, and thus the coding vector v is usually sparse, i.e., the responses are zeros on those codewords which are not chosen.

2.1 Voting-Based Coding

In voting-based coding, the probability density distribution of features is described by a histogram. Each bin of the histogram reflects the occurrence frequency of features on a codeword. This idea is intuitive and easy to implement. However, it is quite rough to approximate the probability density distribution with a histogram. Two kinds of voting-based coding methods are introduced as follows:

Hard voting [6] assigns each feature to its closest codeword and the coding representation of a feature x is

$$v(i) = \begin{cases} 1, & if\ i = \arg\min\limits_j(\|x - b_j\|_2) \\ 0, & otherwise \end{cases}, i = 1, 2, \ldots, M. \quad (1)$$

Soft voting (SV) [30], [39] describes a feature by multiple codewords using a kernel function (e.g., the Gaussian function) of the distance between features and codewords. The coding representation of a feature x is

$$v(i) = \frac{\exp\left(\|x - b_i\|_2^2/\sigma\right)}{\sum_{k=1}^{K}\exp\left(\|x - b_k\|_2^2/\sigma\right)}, i = 1, 2, \ldots, M, \quad (2)$$

where $\sum_{k=1}^{K}\exp(\|x - b_k\|_2^2/\sigma)$ is the normalization factor, and σ is a smooth parameter. $K = M$ in the original soft voting [30]. In a recent paper [40], K is set to a smaller number and accordingly $[b_1, \ldots, b_K]$ denote the closest K codewords of x. This strategy is demonstrated to be more discriminative in the classification tasks.

Soft voting possesses two advantages over hard voting. First, it uses the kernel function of distance as the coding representation instead of the simple one/zero response in hard voting. Second, multiple codewords are employed for coding rather than the hard assignment (i.e., only using the closest codeword). These two changes are useful to enhance the accuracy of probability density estimation.

2.2 Fisher Coding

Fisher coding [31] is inspired by the technique of Fisher kernel, which describes a signal with a gradient vector derived from its probability density function [41]. The gradient vector indicates the direction in which parameters should be adjusted to best fit the data. In the context of image classification, the signal is an image and the gradient vector is used for feature coding. After the original Fisher coding [31] was proposed, there are some extended [32], [42], [43] or simplified [44], [45], [46] versions of Fisher coding. Here, we take the improved Fisher kernel [32] as an example, who achieves the best performance to the best of our knowledge.

In IFK, the probability density distribution of features is described by the Gaussian mixture models. The parameters of GMM, i.e., $\theta_m = \{\omega_m, \mu_m, \Sigma_m\}$, denote the weight, the mean vector, and the covariance matrix of the mth Gaussian distribution, which can be generally estimated by the expectation maximization (EM) algorithm [47].

Supposing that all features are independent each other, an image can be expressed as the log likelihood of all extracted features:

$$L(\mathcal{X} \mid \theta) = \sum_{n=1}^{N} \log\ p(x_n \mid \theta), \quad (3)$$

where $p(x_n \mid \theta)$ is the GMM-based probability density function. The normalized gradient vector, called the Fisher vector, is represented as

$$\mathcal{G} = F_\theta^{-1/2} G, \quad (4)$$

where $G = \nabla_\theta L(\mathcal{X} \mid \theta) = [\partial L/\partial\mu, \partial L/\partial\Sigma]^1$ and F_θ is the Fisher information matrix calculated as

$$F_\theta = E_{\mathcal{X}_p,\mathcal{X}_q}(\nabla_\theta L(\mathcal{X}_p \mid \theta)\nabla_\theta L(\mathcal{X}_q \mid \theta)), \quad (5)$$

where $\mathcal{X}_p$ and $\mathcal{X}_q$ denote two sets of features extracted from two arbitrary images.

The Fisher information has an approximated close solution according to [31], with which the coding vector of a feature, i.e., the Fisher vector, can be represented as the

$$\begin{aligned} &v(i) = [\mathcal{G}_{\mu,i}; \mathcal{G}_{\Sigma,i}], i = 1, 2, \ldots, M, \\ &\mathcal{G}_{\mu,i} = r_i \Sigma_j^{-1/2}(x - \mu_i)/\sqrt{\omega_i}, \\ &\mathcal{G}_{\Sigma,i} = r_i\big((x - \mu_i)\Sigma_i^{-1}(x - \mu_i) - 1\big)/\sqrt{2\omega_i}, \\ &r_i = \omega_i p_i(x \mid \theta)\Big/\sum_{j=1}^{M}\omega_j p_j(x \mid \theta). \end{aligned} \quad (6)$$

1. The derivative to ω, according to [32], makes little contribution to the performance. Thus, it is removed in IFK.

TABLE 1
Three Constraint Functions in Different Reconstruction-Based Coding Algorithms

Coding methods	$\phi(v)$
Sparse coding [25]	$\sum_{i=1}^{M} \lvert v(i)\rvert$
LCC [26]	$\sum_{i=1}^{M} \lvert v(i)\rvert \lVert x-b_i\rVert_2^2$
LLC [33]	$\sum_{i=1}^{M}(v(i)\exp(\lVert x-b_i\rVert_2/\sigma))^2$

2.3 Reconstruction-Based Coding

The core idea of reconstruction-based coding is to reconstruct a feature with codewords via resolving a least-square-based optimization problem with constraints on the codewords. The unified representation of reconstruction-based coding can be generally written as

$$\begin{aligned} &\arg\min_{v} \|x - v\mathcal{B}^T\|_2^2 + \lambda\phi(v) \\ &s.t. \sum_{i}^{M} v(i) = 1, \end{aligned} \tag{7}$$

where the least-square term $\|x - v\mathcal{B}^T\|_2^2$ pursues accurate reconstruction, i.e., a feature can be described with a small error, and the constraint term $\phi(v)$ pursues discriminative description, i.e., similar/different features obtain similar/different representations. The reconstruction coefficients v are used as the coding vector of feature x. The main difference among various reconstruction-based coding methods lies in the constraint term. Three constraint functions are listed in Table 1 as examples, and their meanings will be explained in Section 3.2.

Reconstruction-based coding has been very hot because sparse coding was applied for image classification [25]. In addition to the three methods listed in Table 1, there are many other reconstruction-based coding methods in the recent literature, such as Laplacian sparse coding [48], mixture sparse coding [49], discriminative affine sparse coding [50], nonnegative sparse coding [51], multilayer group sparse coding [52], hierarchical sparse coding [53], and weakly supervised sparse coding [54]. All of them extend sparse coding by substituting the constraint term. Due to the limited space, we do not introduce them one by one here.

2.4 Local Tangent-Based Coding

Local tangent-based coding [34] assumes that all features constitute a smooth manifold where codewords are also located. Feature coding is then interpreted as manifold approximation using the codewords. In this way, features are not independent but closely related, expressed by a Lipschitz smooth function. The main components in local tangent-based coding are manifold approximation and intrinsic dimensionality estimation, which are to be introduced, respectively, as follows:

Denote $f(x)$ as the Lipschitz smooth function of the feature manifold, and it can be described by a high-order representation:

$$|f(x) - f(\tilde{x}) - 0.5(\nabla f(x) + \nabla f(\tilde{x}))^T(x - \tilde{x})| \le \nu\|x - \tilde{x}\|_2^3, \tag{8}$$

where $\tilde{x}$ is a linear combination of several codewords, i.e., $\tilde{x} = \sum_{b\in\mathcal{B}} \gamma_b b$, and ν is the Lipschitz Hessian constant of $f(x)$. The feature manifold can be approximated as

$$f(x) \approx \sum_{b\in\mathcal{B}} \big(\gamma_b f(b) + 0.5\gamma_b \nabla f(b)^T \gamma_b(x - b)\big) \tag{9}$$

with a third-order error $O(\|x - b\|_2^3)$. The derivation from (8) to (9) is provided in the end of this section.

In (9), $f(x)$ is described in the original feature space because $\gamma_b(x - b)$ has the same dimensionality as the feature space. To obtain the intrinsic dimensionality of the feature manifold, PCA is applied over the weighted training data $\gamma_b(x_i - b)$ to solve the projection matrix $U = [u_1(b), u_2(b), \ldots, u_C(b)]$, i.e., the local tangent directions of the manifold. With the projection matrix, (9) is decomposed into

$$f(x) \approx \sum_{b\in\mathcal{B}} \left(\gamma_b f(b) + 0.5\sum_{k=1}^{C} \gamma_b \nabla f(b)^T u_k(b)(x - b)^T u_k(b)\right). \tag{10}$$

In this way, the dimensionality of the feature representation is reduced from D to C.

In local tangent-based coding, only a part of the manifold representation is fed into a linear classifier. This part is easy to be calculated and used as the representation of feature coding:

$$v = \left[\gamma_b; \gamma_b(x - b)^T u_k(b)\right]_{b\in\mathcal{B},k=1,2,\ldots,C}, \tag{11}$$

where γ_b and $u_k(b)$ are calculated by applying LCC and PCA, respectively. The rest part, i.e., $f(b)$ and $\nabla f(b)^T u_k(b)$, can be solved by the linear classifier.

Finally, we provide the derivation details from (8) to (9). As $f(x)$ is Lipschitz smooth, for all $x \in \mathbb{R}^D$:

$$f(\tilde{x}) = f\left(\sum_{b\in\mathcal{B}} \gamma_b b\right) = \sum_{b\in\mathcal{B}} \gamma_b f(b), \tag{12}$$

$$0.5\nabla f(x)^T(x - \tilde{x}) = 0.5\alpha\left(x - \sum_{b\in\mathcal{B}} \gamma_b b\right), \tag{13}$$

$$\begin{aligned} 0.5\nabla f(\tilde{x})^T(x - \tilde{x}) &= 0.5\nabla f\left(\sum_{b\in\mathcal{B}} \gamma_b b\right)^T \left(x - \sum_{b\in\mathcal{B}} \gamma_b b\right) \\ &= 0.5\sum_{b\in\mathcal{B}} \gamma_b \nabla f(b)^T \gamma_b(x - b), \end{aligned} \tag{14}$$

$$\nu\|x - \tilde{x}\|_2^3 = v\|x - \sum_{b\in\mathcal{B}} \gamma_b b\|_2^3 = O\big(\|x - b\|_2^3\big). \tag{15}$$

Substitute (12) $\sim$ (15) into (8) and obtain

$$\begin{aligned} &\left| f(x) - \sum_{b\in\mathcal{B}} \left(\gamma_b f(b) - 0.5\sum_{b\in\mathcal{B}} \gamma_b \nabla f(b)^T \gamma_b(x - b)\right)\right| \\ &\quad \le 0.5\alpha\left(x - \sum_{b\in\mathcal{B}} \gamma_b b\right) + O\big(\|x - b\|_2^3\big). \end{aligned} \tag{16}$$

The difference between x and $\sum_{b\in\mathcal{B}}\gamma_b b$ can be very small if we choose x's nearby codewords to calculate $\sum_{b\in\mathcal{B}}\gamma_b b$. Therefore, (16) can be written as (9) with a third-order error $O(\|x-b\|_2^3)$.

2.5 Saliency-Based Coding

The core idea of saliency-based coding [36] is that saliency is one of the fundamental characteristics of feature coding when combining with MAX pooling. In saliency-based coding, a strong response on a codeword indicates *relative proximity* (corresponding to saliency representation), which means that this codeword, compared with all other codewords, is *much* closer to a feature belonging to this codeword. As a result, the codeword can independently describe this feature without the help of other codewords. Considering that only the strongest response is preserved in the subsequent MAX pooling, relative proximity is more stable than absolute proximity.

The original salient coding employs the difference between the closest codeword and the other $K-1$ closest codewords to reflect saliency, and a feature is accordingly represented as

$$\begin{aligned} v(i) &= \begin{cases} \psi(x), & if\ i = \arg\min_j(\|x-b_j\|_2) \\ 0, & otherwise, \end{cases} \\ \psi(x) &= \sum_{j=2}^{K}(\|x-\tilde{b}_j\|_2 - \|x-\tilde{b}_1\|_2)/\|x-\tilde{b}_j\|_2, \end{aligned} \tag{17}$$

where $\psi(x)$ denotes the saliency degree and $[\tilde{b}_1,\tilde{b}_2,\ldots,\tilde{b}_k]$ is the K closest codewords to x.

It is generally considered that saliency is inherently an exclusive characteristic according to the definition of relative proximity. That is, only the closest codeword is closer to the feature than all other codewords. Therefore, in the original saliency-based coding, hard assignment is used (see (17)). However, hard assignment is a coarse method for feature description. Very recently, Wu et al. [37] propose group saliency-based coding (GSC) by introducing group coding. Its idea is to calculate the saliency response of a group of codewords, and the response is then fed back to all the codewords in the group. The final coding result of a feature on each codeword is the maximum of all responses calculated according to different group sizes.

Let s_i^k denote the ith entry of the coding result obtained with the group size k, $\psi^k(x)$ denote a function measuring the group saliency degree, and $g(x,k)$ denote the set of the k closest codewords of x. In group saliency coding, a feature is represented as

$$\begin{aligned} v(i) &= \max_k\{s_i^k\}, k=1,\ldots,K, \\ s_i^k &= \begin{cases} \psi^k(x), & if\ b_i \in g(x,k) \\ 0, & otherwise, \end{cases} \\ \psi^k(x) &= \sum_{j=1}^{K+1-k}\|x-\tilde{b}_{k+j}\|_2 - \|x-\tilde{b}_k\|_2, \end{aligned} \tag{18}$$

where K is the maximum group size.

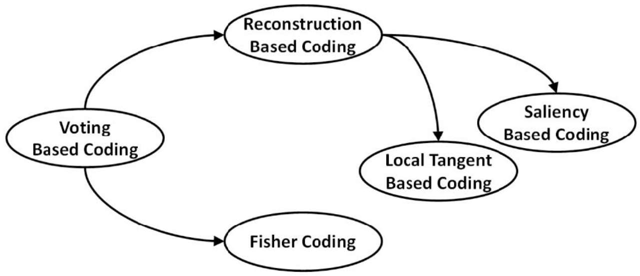

Fig. 3. An evolution map of feature coding.

3 Theoretical Study

In this section, we provide our understanding on the connections among different kinds of coding algorithms based on their characteristics introduced in Section 2. According to the analysis to be given subsequently, the evolution map of feature coding methods is illustrated in Fig. 3, and each step of the evolutional relations is detailed in the following sections, respectively.

3.1 From "Voting" to "Fisher Coding"

Both voting-based coding and Fisher coding aim to describe the whole feature space. Their main difference is the way of describing the probability density distribution of features.

As we analyzed previously, in voting-based coding, the probability density distribution is described by a histogram. Each bin of the histogram corresponds to the occurrence information of a codeword. However, the original dimensionality of the codeword representation is high, for example, 128 for SIFT features. Using one value for a codeword would ignore some useful information.

In Fisher coding, features are described by multiple high-dimensional Gaussian distributions, each of which corresponds to a codeword. Compared with histogram description, GMM contains much richer information. In particular, voting-based coding can be approximated as a simplified version of Fisher coding. We take soft voting as an example for analysis. The probability representation of a feature with GMM is

$$p(x \mid \theta) = \sum_{m=1}^{M}\omega_m p_m(x \mid \theta), \tag{19}$$

$$p_m(x \mid \theta) = \frac{\exp\left(-\frac{1}{2}(x-\mu_m)^T\Sigma_m^{-1}(x-\mu_m)\right)}{(2\pi)^{D/2}|\Sigma_m|^{1/2}}, \tag{20}$$

where $\theta = \{\omega_m, \mu_m, \Sigma_m\}$ denote the weight, the mean vector, and the covariance matrix of the mth Gaussian distribution, and $p_m(x \mid \theta)$ reflects the probability that x belongs to the mth Gaussian distribution.

Soft voting has only one prior parameter, i.e., the codewords $\mathcal{B}$. The probability representation in soft voting (see (2)), can be approximately rewritten as

$$p(x \mid \mathcal{B}) = \sum_{m=1}^{M}p_m(x \mid \mathcal{B}), \tag{21}$$

$$\begin{aligned} p_m(x \mid \mathcal{B}) &= \exp(\|x - b_m\|_2^2/\sigma) \\ &= \exp\left(\sum_{m=1}^{D}(x_d - b_{m,d})^2 \Big/ \sigma\right) \\ &= \prod_{d=1}^{D} \exp((x_d - b_{m,d})^2/\sigma) \\ &\approx \prod_{d=1}^{D} p_{m,d}(x \mid \theta), \end{aligned} \tag{22}$$

where x_d, $b_{m,d}$, and $p_{m,d}(x \mid \theta)$ denote the dth dimension of x, b_m, and $p_m(x \mid \theta)$, respectively.

With the above derivation, it is not hard to draw the following conclusions about the relation between soft voting and Fisher coding:

1. The product operation in (22) indicates that all dimensionalities of the probability representation in soft voting are independent of each other, which is a strong assumption. In Fisher coding, the relations among different dimensionalities are modeled by the joint probability representation with GMM.
2. Fisher coding considers more prior knowledge in probability representation. The weight vector ω in (19) reflects the prior occurrence frequency of features on each Gaussian, which is ignored by soft voting. The covariance matrix Σ in (20) is degenerated into a constant value σ in (22). This means to force the distributions of features on all codewords to be of the same variance, which is also an unreasonable assumption.

Another important difference between Fisher coding and soft voting is that Fisher coding adopts the derivative of the probability density distribution of features (see (3)-(6)), while soft voting only uses PDD itself for feature coding. The use of the derivative is originated from the technique of Fisher kernel [41], which describes a signal with a gradient vector derived from its probability density function. With the use of the derivative, the representation of features can be accordingly adjusted to best approximate data distribution.

Based on the above analysis, it is natural to predict that Fisher coding would perform better than soft voting with the same number of Gaussian distributions/codewords.

3.2 From "Voting" to "Reconstruction"

If we carefully observe the representation of one feature, it is easy to find that reconstruction-based coding achieves more exact description to each feature than voting-based coding. To better understand the relation between voting-based and reconstruction-based coding, we take hard voting and sparse coding for comparison. Without loss of generality, hard voting can be rewritten as

$$\begin{aligned} &\arg\min_v \|x - v\mathcal{B}^T\|_2^2 \\ &s.t.\ \|v\|_0 = 1, \sum_i^M v(i) = 1, \end{aligned} \tag{23}$$

where the l_0-norm counts the number of nonzero entries in a vector. Generally, the constraint $\|v\|_0 = 1$ is considered to be too strong, leading to a rough description to x. In sparse coding, the l_1-norm is adopted and integrated into the objective function:

$$\begin{aligned} &\arg\min_v \|x - v\mathcal{B}^T\|_2^2 + \lambda\|v\|_1 \\ &s.t.\ \sum_i^M v(i) = 1. \end{aligned} \tag{24}$$

With the l_1-norm constraint, sparse coding achieves the effect that similar features share a part of codewords. Further studies (LCC [26]) found that the locality constraint plays a more important role in increasing the probability of such effect. The locality constraint is achieved by minimizing the euclidean distance between a feature and codewords. To model this constraint, $\|v\|_1$ used in sparse coding is replaced with $\sum_i |v(i)|\|x - b(i)\|_2^2$. In this way, LCC focuses on features' nearby codewords that are more likely to be shared by similar features.

The computational cost of LCC is high because its solution relies on iterative optimization. To address this problem, LLC [33] adopts a new constraint function $\sum_i (v(i)\exp(\|x - b_i\|_2/\sigma))^2$. The main difference is that $|v(i)|$ is changed to differentiable $v(i)^2$, so as to obtain an analytical solution in encoding a feature:

$$\begin{aligned} v &= \tilde{v}/1^T\tilde{v}, \\ \tilde{v} &= \left((\mathcal{B} - x)(\mathcal{B} - x)^T + \lambda diag(Dis)\right) \setminus 1, \\ Dis &= \exp(\|x - \mathcal{B}\|_2/\sigma). \end{aligned} \tag{25}$$

The above idea is also presented in [25]. To further enhance the coding speed, approximated LLC is proposed in [33], wherein the constraint function is replaced by using the K closest codewords, corresponding to the following problem:

$$\begin{aligned} &\arg\min_v \|x - v\tilde{\mathcal{B}}^T\|_2^2 \\ &s.t.\ \sum_i^K v(i) = 1, \end{aligned} \tag{26}$$

where $\tilde{\mathcal{B}}$ is the K closest codewords of x. On the one hand, since $\tilde{\mathcal{B}}$ is the K closest codewords, approximated LLC achieves locality. On the other hand, as K is usually a much smaller number compared to the number of codewords, approximated LLC also obtains sparsity.

Further, we exploit the relation between hard voting and approximated LLC. The coding representation of approximated LLC, i.e., (26), can be approximately rewritten as

$$\begin{aligned} &\arg\min_v \left\|x - v\mathcal{B}^T\right\|_2^2 \\ &s.t.\ \|v\|_0 = K, \sum_i^M v(i) = 1. \end{aligned} \tag{27}$$

This form is very similar to that of hard voting (23), which demonstrates that hard voting can be considered as a special case of approximated LLC, i.e., when $K = 1$. Based on the above analysis, it is expected that reconstruction-based coding should perform better than voting-based coding.

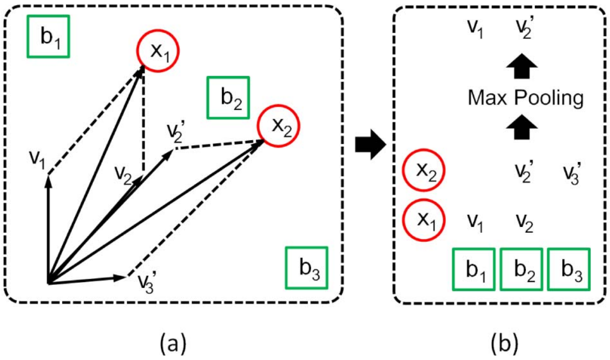

Fig. 4. An illustration of the joint effect of reconstruction and MAX pooling. (a) Least-square-based reconstruction. (b) Coding result after MAX pooling. x_1 and x_2 are two features. $b_1 \sim b_3$ are three codewords. v_1 and v_2 are the responses of x_1 on b_1 and b_2. v_1' and v_2' are the responses of x_2 on b_2 and b_3.

3.3 From "Reconstruction" to "Saliency"

Reconstruction-based coding adopts the least-square-based optimization to generate feature description. The least-square-based optimization is usually an underdetermined system, in which the dimensionality of x is larger than the number of the codewords used for reconstruction. Therefore, it is almost inevitable for the least-square-based optimization to induce deviations in reconstruction. In spite of this, it still achieves surprisingly good performance in some databases. The secret may lie in the salient representation when combining with MAX pooling. Take approximated LLC as an example. As each codeword may be used multiple times in reconstructing features, it may receive multiple responses. However, in the later MAX pooling, only the maximum response is preserved. What is the meaning of these maximum responses? We illustrate it in Fig. 4.

Fig. 4a depicts the geometric explanation of reconstruction in a 2-dimensional feature space in the case of $K = 2$. In approximated LLC, the reconstruction in (26) is an analog of vector composition following the parallelogram law [55]. When a feature, for example, x_2, is close to b_2 and far away from b_3, approximated LLC produces a strong response v_2' on b_2 and a weak response v_3' on b_3. When a feature, for example, x_1, is located in the middle of two codewords, for example, b_1 and b_2, both v_1 and v_2 are relatively weak. For the case with a larger K, the analysis is similar, i.e., using the parallelogram law multiple times.

What is the underlying meaning of Fig. 4? When a codeword obtains a strong response, i.e., it is much closer to a feature compared with other codewords, this codeword can *independently* describe the feature (*salient representation*). This is the case of b_2, b_3, and x_2 in Fig. 4a, where v_2' can approximately represent x_2 without v_3'. When all responses in representing a feature are weak (*unsalient representation*), all related codewords should be preserved to describe this feature. This is the case of b_1, b_2, and x_1, where the response on a single codeword *cannot independently* represent the feature. In this case, the response is *unstable* because a weak response may be suppressed in the subsequent MAX pooling, for example, v_2 is suppressed by v_2' (see Fig. 4b). In a word, salient representation leads to stable description.

The least-square-based reconstruction can obtain salient representation in a low-dimensional feature space, which is guaranteed by the parallelogram law. However, if K is smaller than the dimensionality of the feature space, features and codewords may not be in the same hyperplane, and thus, exact reconstruction cannot be achieved by using K codewords. Fig. 5 illustrates an example wherein b_1 and b_2 in the 3-dimensional space cannot represent x by the parallelogram law. The least-square-based optimization uses b_1 and b_2 to reconstruct x and accordingly the coding response is the projection vector of x in the plane determined by b_1 and b_2. The distance between x and x' is the deviation of reconstruction.

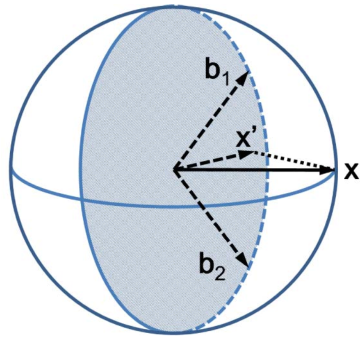

Fig. 5. An example showing the deviation generated during least-square-based reconstruction.

After recognizing the importance of salient representation and the problem of least-square-based reconstruction, saliency-based coding [36] is proposed to directly extract salient representation according to the difference between the closest codeword and other $K-1$ codewords. The underlying intuition is that the larger the difference, the more salient the representation. The mathematical representation of original saliency coding is listed in (17). Further, group saliency coding is presented [37] that uses the difference between the closest group of codewords and other codewords to reflect saliency. The spirit of GSC is similar to that of the original saliency coding, and the main difference is that GSC considers saliency representation on multiple codewords but not a single codeword, which avoids the possible rough description induced by hard assignment. The mathematical representation of group saliency coding is listed in (18).

Compared with reconstruction-based coding, saliency-based coding owes two advantages: 1) directly derived from the definition of saliency without the underdetermined problem in the least-square-based reconstruction; and 2) easy to implement without iterative optimization and, thus, performs much faster.

3.4 From "Reconstruction" to "Local Tangent"

Reconstruction-based coding employs least-square-based optimization, which actually builds a local linear approximation for every feature. However, this strategy, due to its under-determined solution, will inevitably generate reconstruction deviation, as illustrated in Fig. 5. Local tangent-based coding is another way that also pursues exact description to each feature. The main difference between reconstruction-based coding and local tangent-based coding is the estimation to the feature manifold.

Take LCC as an example. Similar to (8), the feature manifold in LCC, according to [34], can be expressed as

$$\left|f(x) - \sum_{b \in \mathcal{B}} \gamma_b f(b)\right| \leq 0.5\alpha\left(x - \sum_{b \in \mathcal{B}} \gamma_b b\right) + \beta \sum_{b \in \mathcal{B}} |\gamma_b| \|x - b\|_2^2. \tag{28}$$

Accordingly, the manifold estimation in LCC is

$$f(x) \approx \sum_{b \in \mathcal{B}} \gamma_b f(b), \tag{29}$$

with a second-order error $O(\|x - b\|_2^2)$.

Comparing (29) and (9), it is clear to see that LCC uses a *linear* combination and local tangent-based coding employs a *nonlinear quadratic* function to approximate the feature manifold. The approximation error decreases from $O(\|x - b\|_2^2)$ in LCC to $O(\|x - b\|_2^3)$ in local tangent-based coding. In other words, local tangent-based coding describes the feature manifold more preciously [34].

From another viewpoint, local tangent-based coding can be considered as a reconstruction without deviation. To explain this idea, we first introduce super-vector coding [35], a simplified version of local tangent-based coding. Its core idea is simplifying (10) to

$$f(x) \approx f(b^*(x)) + 0.5\nabla f(b^*(x))^T (x - b^*(x)), \tag{30}$$

where $b^*(x)$ is the closest codeword of x. Accordingly, (11) is replaced with the so-called super-vector coding:

$$v = [s; (x - b^*(x))], \tag{31}$$

where s is a predefined parameter. Compared with the original local tangent coding in (11), super-vector coding makes two simplifications. First, it only uses the closest codewords of a feature, and thus does not need to run LCC to obtain the coefficient γ_b in (11). Second, the super vector $x - b^*(x)$ is adopted to replace the local tangent vector $(x - b)^T u_k(b)$ in (11).

According to (31), the representation of each feature in super-vector coding can be divided into two parts: $[v_1; v_2] = [s; (x - b^*(x))]$, and a feature x can be represented as

$$x = b^*(x) + v_2. \tag{32}$$

As $b^*(x)$ and v_2 have the same dimensionality as the feature space, x can be reconstructed without deviation, which supports our claim that super-vector coding is an enhanced version of reconstruction-based coding.

3.5 Summary

As the final part of this section, we summarize the characteristics of the five kinds of coding methods in Table 2 and explain them as follow.

Robustness is defined here as insensitiveness to unusual features, for example, noisy features. Global coding pursues to model the probability density distribution of features, and thus, it is not easy to be influenced by a small number of unusual features. In particular, Fisher coding uses GMM for probability density estimation, which is more robust than the histogram-based manner. Local coding aims to describe each individual feature and, thus, is sensitive to unusual features.

As the codebook size increases, local coding can describe more patterns of features. Therefore, it has good *adaptiveness* to the increase of the codebook size. In contrast, in global coding, there should be an optimal codebook size (corresponding to the best division of the feature space) to estimate the probability density distribution.

TABLE 2
Characteristics of Coding Methods

	Robustness	Adaptiveness	Accuracy	Independency
Voting	✓	–	–	–
Fisher Coding	✓✓	–	–	–
Reconstruction	–	✓	✓	✓
Saliency	–	✓	–	✓✓
Tangent	–	✓	✓✓	✓

Reconstruction-based coding and local tangent-based coding pursue exact description to each feature and, thus, perform well in the *accuracy* of feature reconstruction. In particular, SVC reconstructs a feature without deviation and outperforms others in accuracy.

Independency means that a codeword can stably represent a pattern of features. Local coding is designed to describe each feature and, thus, performs better than global coding in this aspect. Especially, saliency-based coding proposes salient representation in which the preserved strong response on each codeword can independently express a feature without the help of other codewords.

4 Experimental Study

To test the performance of different kinds of coding methods, an empirical study is conducted in this section. First, we introduce used databases and experimental setup, then choose five representative coding methods for extensive evaluation, followed with the result analysis.

4.1 Experimental Databases

Four databases are chosen for empirical evaluation, which are, respectively:

- The 15-Scenes data set [8] is a typical database for scene classification. It consists of 4,485 images spread over 15 categories, each of which contains 200 to 400 images. We follow the experimental setup of Lazebnik et al. [56] wherein 100 random images per class are chosen as training samples and the rest are used for testing.
- The Caltech-256 database [9] is a typical database for object classification. It consists of 29,780 images including 256 object categories plus a background class. Each category contains at least 80 images. We use the common experimental setting on this database: For training, using different numbers of images; for testing, randomly choosing at most 25 images per class.
- The PASCAL VOC07 database [57] is one of the most challenging databases for image classification with 9,963 images distributed in 20 classes of objects. All images are obtained from Flickr with large variations in size, illumination, scale, viewpoint, deformation, and clutter. The training and testing samples

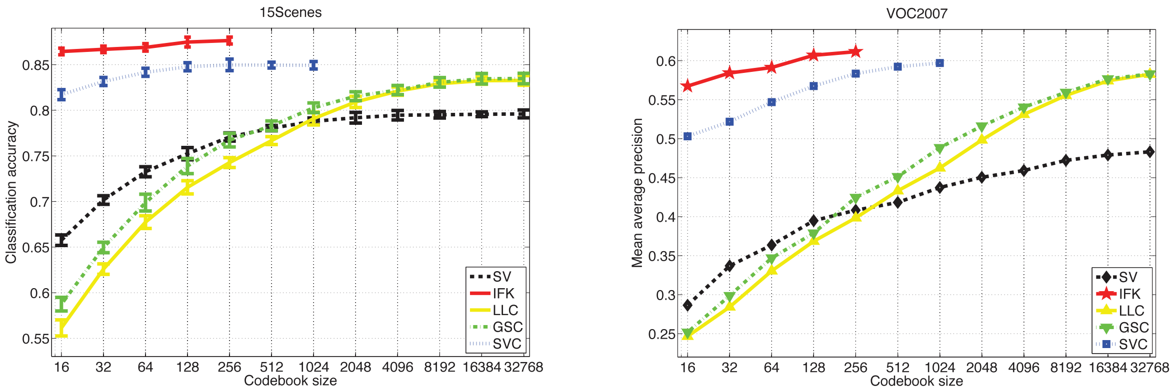

Fig. 6. Performance comparison on the 15-Scenes and the PASCAL VOC07 databases.

have been well divided. PASCAL VOC07 is the latest one of the PASCAL VOC Challenge data sets with the labels of the testing images released, and thus is convenient for evaluation.

- The SUN397 database [58] is probably the largest database for scene classification. It contains 108,754 images over 397 well-sampled categories. The number of images varies across categories, but there are at least 100 images per category. Ten subsets of the data set have been chosen for evaluation, each of which has 50 training images and 50 testing images per class. We follow the common experimental setting [58] on this database. In each experiment, different number of images are used for training, and all the 50 testing images are used for testing no matter what size the training set is.

4.2 Experimental Setup

In all experiments, we adopt the 128-dimensional SIFT feature [16] densely extracted from images on a grid with a step of 4 pixels under three scales: $16 \times 16, 24 \times 24$, and 32×32, using the released code from VLFeat [59]. To generate codewords, we use the standard K-means clustering algorithm [21] for all coding methods except for Fisher coding, wherein the GMM is applied. After all features are encoded, spatial pyramid matching (SPM) [56] is performed following most previous work. That is, on the Caltech-256 database and the SUN397 database, SPM with levels of $[1 \times 1, 2 \times 2, 4 \times 4]$ is used. On the 15-Scenes database and the PASCAL VOC07 database, SPM with levels of $[1 \times 1, 2 \times 2, 3 \times 1]$ is employed. All coding methods keep the same pooling operations used in their original literature, which is consistent with previous work. That is, soft voting and Fisher coding are combined with average pooling, super-vector coding is combined with weighted average pooling, and others are combined with MAX pooling. For normalization, we l_2-normalizes the square root of the responses. In training and testing, Lib-linear SVM [60] is adopted, wherein the penalty coefficient is determined via cross validation.

On 15-Scenes, Caltech-256, and SUN397, following most previous work, we repeat the experiment 10 times and report the average accuracy and the standard deviation. On the VOC07, the performance is measured with the mean average precision (MAP), used in the PASCAL VOC competition [10].

4.3 Selection of Coding Methods

According to Fig. 3, we choose five coding methods:

- Soft voting [61] is chosen as the representative of voting-based coding methods. It adopts soft assignment and can obtain more accurate probability density estimation than hard voting.
- Improved Fisher Kernel [32] is the representative of Fisher coding. It improves the original Fisher coding [31] and achieves the best performance in [38].
- Local-constrained linear coding (LLC) [33] is chosen as the representative of reconstruction-based coding because it is much faster than most reconstruction-based coding methods. Meanwhile, it performs better than some classic methods such as sparse coding [25].
- Group saliency coding [37] is the newest study of saliency-based coding, which avoids the problem of hard assignment in the original salient coding [36].
- Super vector-coding [35] inherits the main characteristics of the original local tangent-based coding [34] and runs much faster.

4.4 Analysis of Results

The tendencies of performance curves on four databases (Figs. 6, 7, and 8) are similar. We explain main experimental findings in the following:

1. Experimental figures on four databases basically justify the correctness of the proposed evolutional directions: with the same codewords, Fisher coding and reconstruction-based coding outperform voting-based coding; Local tangent coding and saliency-based coding perform better than reconstruction-based coding. Overall, Fisher coding performs best. We believe this is probably because *robustness* plays the most important role among the four characteristics shown in Table 2. This finding demonstrates that it is possible that objects belonging to the same class contain a number of different local features, even on the same codeword. Fisher coding, due to its excellent robustness, captures this characteristic and, thus, achieves a proper tolerance to unusual local features. To study the robustness of different coding algorithms, we design an additional experiment in which random noises in different proportions are added to replace the original SIFT features.

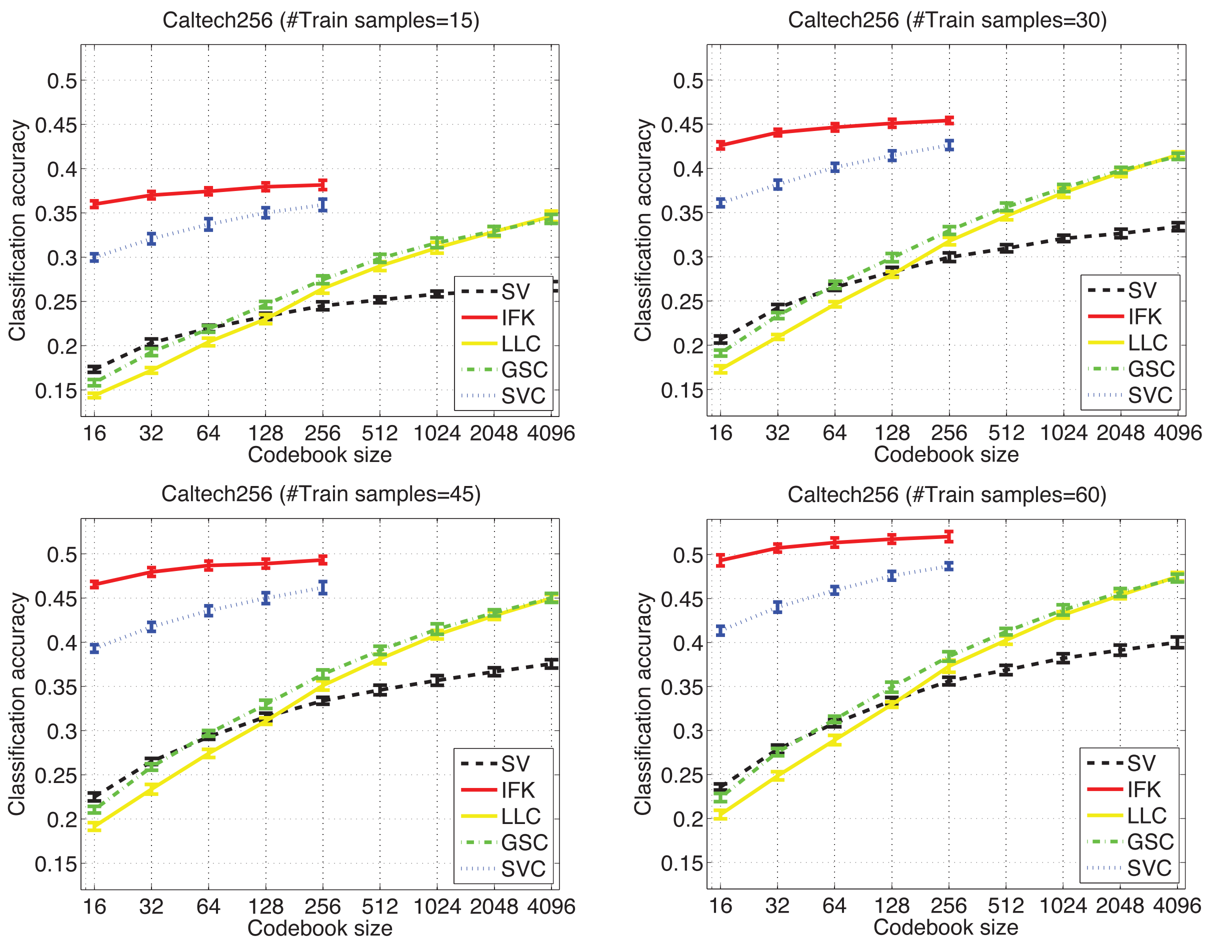

Fig. 7. Performance comparison on the Caltech-256 database. Please note that the maximal codebook size of SVC in this figure decreases to 256 because images on this database are much more than those on 15-Scenes and VOC07.

The experimental results are illustrated in Fig. 9. It is clear that the mean average precision of IFK decreases most slowly, showing that it is more robust to noises. This result also supports our claim that robustness is an important factor that helps IFK perform well.

2. Influence of the codebook size. The overall tendency is: the more the codewords, the better the performance. However, there is an overfitting effect when the dimensionality of the coding representation becomes very large, which leads to the plateau of performance curves. For example, on the 15-Scenes database, there are nearly no increases for LLC and GSC when the codebook size arrives at 16,384. For SV, there is even a slight decrease after 4,096. However, on the PASCAL VOC07 and the Caltech-256 databases, the overfitting effect is not obvious. This is possibly because these two databases have relatively high tolerance to overfitting. We believe that the performance will deteriorate if we use a larger size of the codebook. To justify it, we add an experiment with an extremely high dimensionality on the PASCAL VOC07 database, shown in Fig. 10. The result justifies our prediction. The performance of SV and LLC decreases after 65,536.
3. Influence of the number of training samples. The experimental results on the Caltech-256 and SUN397 databases clearly show that the increase of training samples consistently enhances the performance of all coding methods.
4. The similar tendency in Figs. 6, 7, and 8 indicates some useful guidelines in practical applications:

   a. SV is a good choice for the case of high speed and low memory cost;
   b. IFK is suitable to pursue high classification accuracy; and
   c. GSC can be taken into account for the balance between speed and accuracy.

Besides, we carefully compare our experimental results with Chatfield et al. [38] experimental evaluation, which is widely accepted and cited. However, for SV, they do not test the case of using the linear kernel, and thus, we compare our result of SV with that implemented by Boureau et al. [29]:

- *SV*. The performance of SV with our implementation is a little better than that reported by Boureau et al. [29]. For example, on the 15-Scenes database, when the codebook size is 1,024, our result and theirs are, respectively, $78.8 \pm 0.4\%$ and $75.6 \pm 0.5\%$ using the linear SVM. The difference is reasonable, considering that our experiment adopts more denser feature sampling rate (every 4 pixels) than theirs (every 8 pixels).

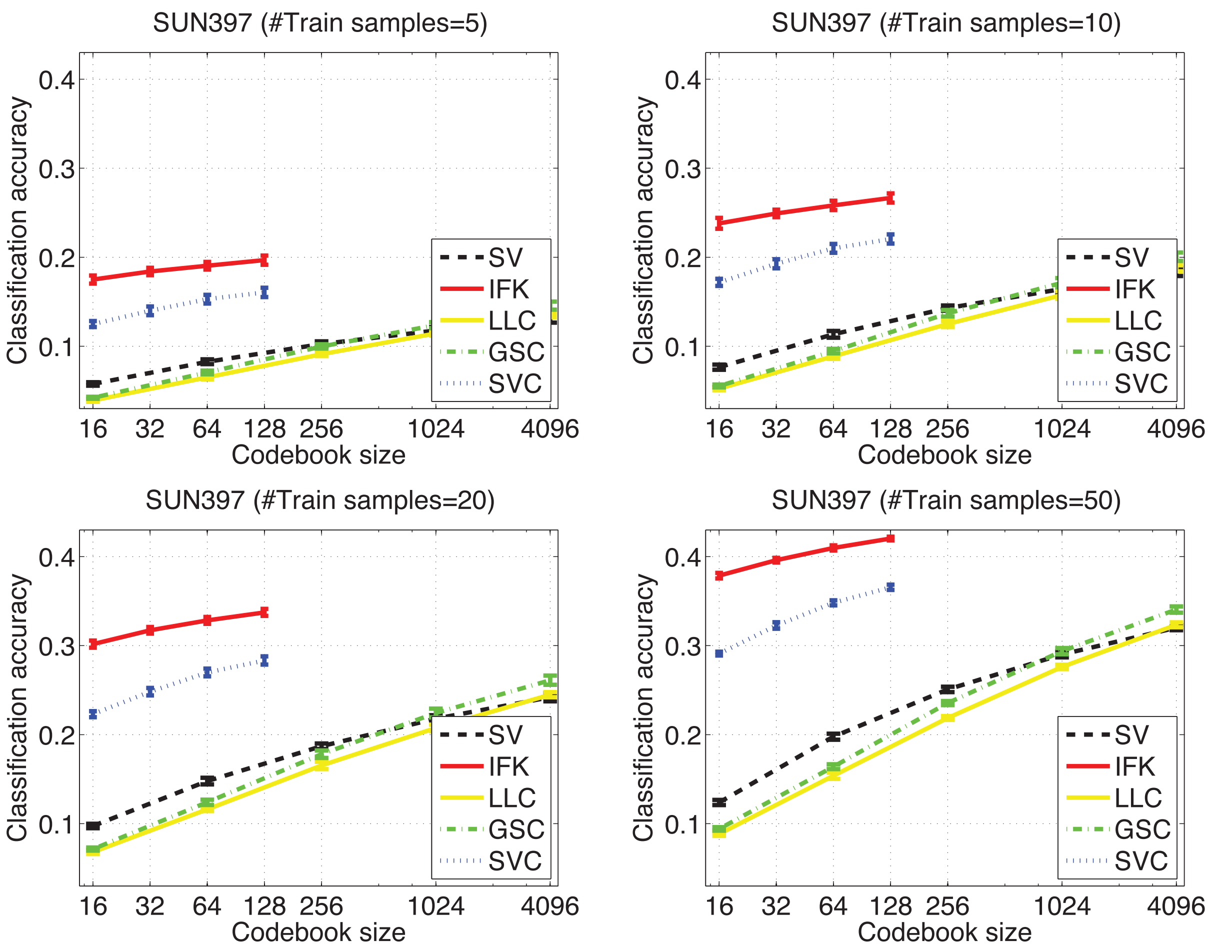

Fig. 8. Performance comparison on the SUN397 database.

- *IFK.* The performance of IFK implemented by us is a little lower than that reported by Chatfield et al. [38]. For example, on the VOC07 database, when the codebook size is 256, our result and theirs are, respectively, 61.2 and 61.69 percent. This is probably caused by different implementations of GMM according to our personal communication with Chatfield.
- *LLC.* The performance of LLC implemented by us is similar to that by Chatfield et al. [38]. We list both of them in Table 3.
- *GSC.* We use the same implementation as the original GSC [37]. Therefore, the experimental result is the same as that of [37].
- *SVC.* The performance of SVC implemented by us is a little better than that by Chatfield et al. [38]. For example, on the VOC07 database, when the codebook size is 1,024, our result and theirs are, respectively, 59.7 and 58.16 percent. This may be induced by different clustering results when generating codewords.

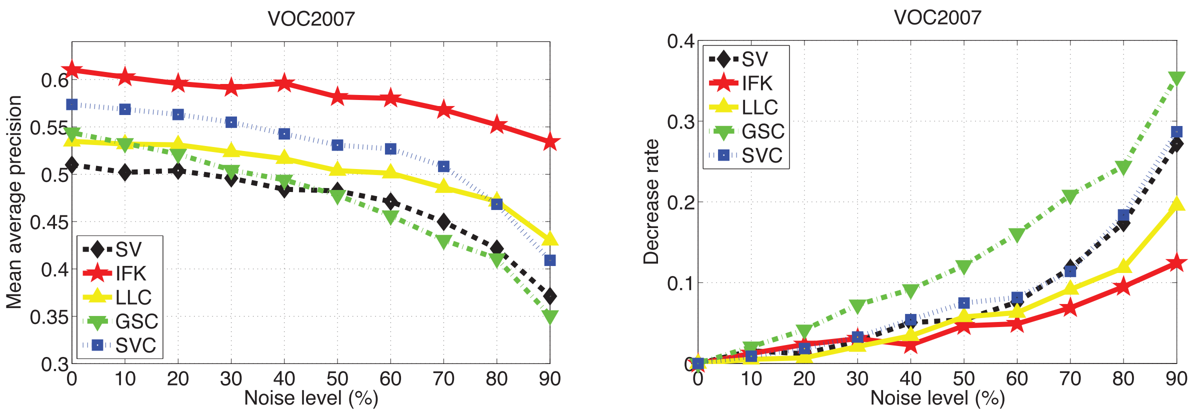

Fig. 9. The influence of random noises on coding algorithms. The baseline algorithms are SV (4,096), IFK (256), LLC (4,096), GSC (4,096), and SVC (256). The numbers in the parenthesis denote the size of the codebook.

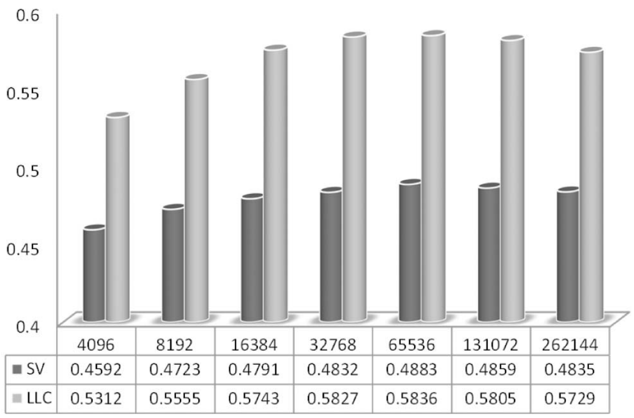

Fig. 10. An illustration of the overfitting effect on the PASCAL VOC07 database. The horizontal and the vertical axes denote the number of codewords and the average precision, respectively. For readers' convenience, the numerical average precision is also listed below the histogram. We only take SV and LLC for justification because this experiment is very time and memory consuming.

TABLE 3
Performance Comparison of Different Implementations of LLC on VOC07

#C/SR	4000/SR3	10000/SR3	14000/SR3	25000/SR3
[38]	53.79%	56.01%	56.18%	57.27%
#C/SR	4096/SR4	8192/SR4	16384/SR4	32768/SR4
Ours	53.12%	55.55%	57.43%	58.27%

"$\#C$" is the number of codewords and "SR" is the sampling rate.

5 Conclusion and Discussion

In this paper, we have discussed various coding methods, including their motivations and mathematical representations. Moreover, we have analyzed their relations in theory, and empirically evaluated their performance. The main conclusions are listed as follows:

1. For global coding, Fisher coding is more reasonable than voting-based coding. First, to estimate the probability density distribution, the high-dimensional GMM used in Fisher coding is more accurate than the histogram used in voting-based coding. Second, Fisher coding takes more prior knowledge into account, for example, the weight and the variance of clusters, and thus it contains richer information.
2. Reconstruction-based coding enhances hard voting in two aspects. First, it uses a linear combination of codewords to approximate features so that the description error is reduced. Second, the constraint on codewords in the objective function (Table 1) leads to the advantage that similar/different features obtain similar/different representations.
3. Saliency-based coding improves reconstruction-based coding via jointly considering coding and pooling. The saliency degree, i.e., the degree that codewords can independently describe features, is a key factor to obtain stable representation.
4. Local tangent-based coding aims to build the local geometry of the feature manifold. The derived feature description can be seen as a kind of feature reconstruction without deviation, and thus, it performs better than traditional reconstruction-based coding.

Finally, we would like to discuss some open directions. Some of them are inspired by the connections among coding methods. Here, we just give several examples:

1. The development from reconstruction-based coding to saliency-based coding tells us that it is important to design feature coding by considering the joint effects of feature coding and pooling. Saliency-based coding exploits the influence of MAX pooling to feature coding. Along this direction, more kinds of pooling, for example, learning-based pooling strategies proposed recently [62], [63], could be jointly considered in the design of feature coding.
2. Voting-based coding estimates the probability density distribution of features using a histogram, while Fisher coding employs more powerful Gaussian mixture models, which achieves much better performance. This relation reveals that accurate probability density estimation of features is critical to enhance the effectiveness of feature coding. Meanwhile, the development from LCC to local tangent coding tells us that the high-order Lipschitz smooth function is available to describe the feature manifold more accurately. Inspired by these theoretical connections, it may be potential to borrow the idea of high-order manifold approximation, as a kind of nonparameter model, to further improve probability density estimation for feature coding.
3. The relations among codewords in feature coding are not explicitly revealed. Here, "explicitly" indicates exploiting the *prior* relations among codewords. Codebook graph [64] is such an attempt, wherein an edge of the graph indicates that two codewords are related. However, the relations among codewords are heuristic in [64]. It is potential to follow this direction for an in-depth study.
4. Most current coding methods focus on the feature space, i.e., the appearance information of features. However, their spatial information is also important, which may be useful to distinguish images of different categories and group similar ones. Very recently, spatially regularized coding, proposed by Shabou and Borgne [65], embeds the spatial distance of features into the constraint term of LLC, increasing the classification accuracy. We believe that spatial modeling with various coding methods will attract much attention in future.

Acknowledgments

This work was jointly supported by the National Natural Science Foundation of China (61135002, 61203252) and the National Basic Research Program of China (2012CB316300), Tsinghua National Laboratory for Information Science and Technology Cross-discipline Foundation, and Hundred Talents Program of CAS. The source code of all coding algorithms and experimental figures in this paper have been released on http://nlpr-web.ia.ac.cn/english/irds/people/yzhuang.html.

REFERENCES

[1] R. Collins, A. Lipton, T. Kanade, H. Fujuyoshi, D. Duggins, Y. Tsin, D. Tolliver, N. Enomoto, and O. Hasegawa, "A System for Video Surveillance and Monitoring," Technical Report CMU-RI-TR-00-12, Carnegie Mellon Univ., Pittsburgh, Penn., 2000.

[2] A. Vailaya, M.A.T. Figueiredo, A.K. Jain, and H.J. Zhang, "Image Classification for Content-Based Indexing," *IEEE Trans. Image Processing*, vol. 10, no. 1, pp. 117-130, Jan. 2001.

[3] R. Kosala and H. Blockeel, "Web Mining Research: A Survey," *ACM SIGKDD Explorations Newsletter*, vol. 2, no. 1, pp. 1-15, 2000.

[4] V.I. Pavlovic, R. Sharma, and T.S. Huang, "Visual Interpretation of Hand Gestures for Human-Computer Interaction: A Review," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 19, no. 7, pp. 677-695, July 1997.

[5] A.K. Jain, A. Ross, and S. Prabhakar, "An Introduction to Biometric Recognition," *IEEE Trans. Circuits and Systems for Video Technology*, vol. 14, no. 1, pp. 4-20, Jan. 2004.

[6] G. Csurka, C. Bray, C. Dance, and L. Fan, "Visual Categorization with Bags of Keypoints," *Proc. ECCV Int'l Workshop Statistical Learning in Computer Vision*, 2004.

[7] T. Joachims, "Text Categorization with Support Vector Machines: Learning with Many Relevant Features," *Proc. 10th European Conf. Machine Learning (ECML '98)*, 1998.

[8] http://www.cs.unc.edu/-lazebnik/research/scene-categories.zip/, 2006.

[9] http://www.vision.caltech.edu/Image-DataSets/Caltech256/, 2013.

[10] http://pascallin.ecs.soton.ac.uk/challenges/voc/, 2012.

[11] http://www.image-net.org/, 2013.

[12] M. Marszalek, C. Schmid, H. Harzallah, and J. van de Weijer, "Learning Representations for Visual Object Class Recognition," *Proc. Visual Recognition Challenge Workshop in Conjunction with IEEE Int'l Conf. Computer Vision*, 2007.

[13] C. Harris and M. Stephens, "A Combined Corner and Edge Detector," *Proc. Fourth Alvey Vision Conf.*, pp. 147-151, 1988.

[14] T. Tuytelaars and L.V. Gool, "Matching Widely Separated Views Based on Affine Invariant Regions," *Int'l J. Computer Vision*, vol. 59, no. 1, pp. 61-85, 2004.

[15] J. Matas, O. Chum, M. Urban, and T. Pajdla, "Robust Wide-Baseline Stereo from Maximally Stable Extremal Regions," *Image and Vision Computing*, vol. 22, no. 10, pp. 761-767, 2004.

[16] D.G. Lowe, "Distinctive Image Features from Scale-Invariant Key-Points," *Int'l J. Computer Vision*, vol. 60, no. 2, pp. 91-110, 2004.

[17] T. Ojala, M. Petikainen, and D. Harwood, "A Comparative Study of Texture Measures with Classification Based on Feature Distributions," *Pattern Recognition*, vol. 29, pp. 51-59, 1996.

[18] N. Dalal and B. Triggs, "Histograms of Oriented Gradients for Human Detection," *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR '05)*, 2005.

[19] K. Mikolajczyk and C. Schmid, "A Performance Evaluation of Local Descriptors," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 27, no. 10, pp. 1615-1630, Oct. 2005.

[20] J.G. Zhang, M. Marszalek, S. Lazebnik, and C. Schmid, "Local Features and Kernels for Classification of Texture and Object Categories: A Comprehensive Study," *Int'l J. Computer Vision*, vol. 73, no. 2, pp. 213-238, 2007.

[21] S.P. Lloyd, "Least Squares Quantization in PCM," *IEEE Trans. Information Theory*, vol. IT-28, no. 2, pp. 129-137, Mar. 1982.

[22] J. Yang, K. Yu, and T. Huang, "Supervised Translation-Invariant Sparse Coding," *Proc. IEEE Conf. Computer Vision and Pattern Recognition(CVPR)*, 2010.

[23] D.M. Bradley and J.A. Bagnell, "Differential Sparse Coding," *Proc. Advances in Neural Information Processing Systems (NIPS)*, 2008.

[24] J. Mairal, F. Bach, J. Ponce, and G. Sapiro, "Supervised Dictionary Learning," *Proc. Advances in Neural Information Processing Systems (NIPS)*, 2008.

[25] J. Yang, K. Yu, Y. Gong, and T. Huang, "Linear Spatial Pyramid Matching Using Sparse Coding for Image Classification," *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, 2009.

[26] K. Yu, T. Wang, and Y. Gong, "Nonlinear Learning Using Local Coordinate Coding," *Proc. Advances in Neural Information Processing Systems (NIPS)*, 2009.

[27] Z. Jiang, G. Zhang, and L. Davis, "Submodular Dictionary Learning for Sparse Coding," *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, 2012.

[28] Y. Boureau, J. Ponce, and Y. Lecun, "A Theoretical Analysis of Feature Pooling in Visual Recognition," *Proc. 27th Int'l Conf. Machine Learning (ICML)*, 2010.

[29] Y. Boureau, F. Bach, Y. LeCun, and J. Ponce, "Learning Mid-Level Features for Recognition," *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, 2010.

[30] J. Gemert, J. Geusebroek, C. Veenman, and A. Smeulders, "Kernel Codebooks for Scene Categorization," *Proc. 10th European Conf. Computer Vision: Part III (ECCV '08)*, 2008.

[31] F. Perronnin and C. Dance, "Fisher Kernels on Visual Vocabularies for Image Categorization," *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, 2007.

[32] F. Perronnin, J. Sanchez, and T. Mensink, "Improving the Fisher Kernel for Large-Scale Image Classification," *Proc. 11th European Conf. Computer Vision: Part IV (ECCV)*, 2010.

[33] J. Wang, J. Yang, K. Yu, F. Lv, T. Huang, and Y. Gong, "Locality-Constrained Linear Coding for Image Classification," *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, 2010.

[34] K. Yu and T. Zhang, "Improved Local Coordinate Coding Using Local Tangents," *Proc. Int'l Conf. Machine Learning (ICML)*, 2010.

[35] X. Zhou, K. Yu, T. Zhang, and T. Huang, "Image Classification Using Super-Vector Coding of Local Image Descriptors," *Proc. 11th European Conf. Computer Vision: Part V (ECCV)*, 2010.

[36] Y. Huang, K. Huang, Y. Yu, and T. Tan, "Salient Coding for Image Classification," *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, 2011.

[37] Z. Wu, Y. Huang, L. Wang, and T. Tan, "Group Encoding of Local Features in Image Classification," *Proc. 21st Int'l Conf. Pattern Recognition (ICPR)*, 2012.

[38] K. Chatfield, V. Lempitsky, A. Vedaldi, and A. Zisserman, "The Devil Is in the Details: An Evaluation of Recent Feature Encoding Methods," *Proc. British Machine Vision Conf. (BMVC)*, 2011.

[39] J. Gemert, J. Geusebroek, C. Veenman, and A. Smeulders, "Visual Word Ambiguity," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 32, no. 7, pp. 1271-1283, July 2010.

[40] L. Liu, L. Wang, and X. Liu, "In Defense of Soft-Assignment Coding," *Proc. IEEE Int'l Conf. Computer Vision (ICCV)*, 2011.

[41] T. Jaakkola and D. Haussler, "Exploiting Generative Models in Discriminative Classifiers," *Proc. Conf. Advances in Neural Information Processing Systems II (NIPS)*, 1999.

[42] F. Perronnin, Y. Liu, J. Sanchez, and H. Poirier, "Large-Scale Image Retrieval with Compressed Fisher Vectors," *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, 2010.

[43] R.G. Cinbis, J. Verbeek, and C. Schmid, "Image Categorization Using Fisher Kernels of Non-Iid Image Models," *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, 2012.

[44] H. Jegou, M. Douze, C. Schmid, and P. Perez, "Aggregating Local Descriptors into a Compact Image Representation," *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, 2010.

[45] H. Jegou, F. Perronnin, M. Douze, J. Sanchez, P. Perez, and C. Schmid, "Aggregating Local Image Descriptors into Compact Codes," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 34, no. 9, pp. 1704-1716, Sept. 2012.

[46] D. Picard and P. Gosselin, "Improving Image Similarity with Vector of Locally Aggregated Tensors," *Proc. 18th IEEE Int'l Conf. Image Processing (ICIP)*, 2011.

[47] G. McLachlan and D. Peel, *Finite Mixture Models.* John Wiley & Sons, 2000.

[48] S. Gao, I. Tsang, L. Chia, and P. Zhao, "Local Features Are Not Lonely—Laplacian Sparse Coding for Image Classification," *Proc. European Conf. Computer Vision (ECCV)*, 2010.

[49] J. Yang, K. Yu, and T. Huang, "Efficient Highly Over-Complete Sparse Coding Using a Mixture Model," *Proc. 11th European Conf. Computer Vision: Part V (ECCV)*, 2010.

[50] K. Naveen and B. Li, "Discriminative Affine Sparse Codes for Image Classification," *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, 2011.

[51] C. Zhang, J. Liu, Q. Tian, C. Xu, H. Lu, and S. Ma, "Image Classification by Non-Negative Sparse Coding, Low-Rank and Sparse Decomposition," *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, 2011.

[52] S. Gao, L. Chia, and I. Tsang, "Multi-Layer Group Sparse Coding for Concurrent Image Classification and Annotation," *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, 2011.

[53] K. Yu, Y. Lin, and J. Lafferty, "Learning Image Representations from the Pixel Level via Hierarchical Sparse Coding," *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, 2011.
[54] L. Cao, R. Ji, Y. Gao, Y. Yang, and Q. Tian, "Weakly Supervised Sparse Coding with Geometric Consistency Pooling," *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, 2012.
[55] http://en.wikipedia.org/wiki/Parallelogram_law, 2013.
[56] S. Lazebnik, C. Schmid, and J. Ponce, "Beyond Bags of Features: Spatial Pyramid Matching for Recognizing Natural Scene Categories," *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, 2006.
[57] http://pascallin.ecs.soton.ac.uk/challenges/VOC/voc2007/index.html, 2013.
[58] J. Xiao, J. Hays, K. Ehinger, A. Oliva, and A. Torralba, "Sun Database: Large-Scale Scene Recognition from Abbey to Zoo," *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, 2010.
[59] http://www.vlfeat.org/, 2013.
[60] http://www.csie.ntu.edu.tw/cjlin/liblinear/, 2013.
[61] J. van Gemert, J. Geusebroek, C. Veenman, and A. Smeulders, "Kernel Codebooks for Scene Categorization," *Proc. 10th European Conf. Computer Vision: Part III (ECCV '08)*, 2008.
[62] J. Feng, B. Ni, Q. Tian, and S. Yan, "Geometric Lp-Norm Feature Pooling for Image Classification," *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, 2011.
[63] Y. Jia, C. Huang, and T. Darrell, "Beyond Spatial Pyramids: Receptive Field Learning for Pooled Image Features," *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, 2012.
[64] Y. Huang, K. Huang, C. Wang, and T. Tan, "Exploring Relations of Visual Words for Image Classification," *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, 2011.
[65] A. Shabou and H. Borgne, "Locality-Constrained and Spatially Regularized Coding for Scene Categorization," *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, 2012.

Yongzhen Huang received the BE degree from Huazhong University of Science and Technology in 2006 and the PhD degree from the Institute of Automation, Chinese Academy of Sciences (CASIA) in 2011. In July 2011, he joined the National Laboratory of Pattern Recognition (NLPR), CASIA, where he is currently an associate professor. He has published more than 30 papers in the areas of computer vision and pattern recognition in international journals and conferences such as the *IEEE Transactions on Systems, Man, and Cybernetics-B*, CVPR, NIPS, ICIP, and ICPR. His current research interests include pattern recognition, computer vision, machine learning, and biologically inspired vision computing. He is a member of the IEEE.

Zifeng Wu received the BSc and MSc degrees from the School of Mechanical Engineering and Automation at Beihang University, Beijing, China, in 2006 and 2009, respectively. He is currently working toward the PhD degree in the National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences, Beijing, China. His research interests include human vision, image classification, and object detection.

Liang Wang received both the BEng and MEng degrees from Anhui University in 1997 and 2000, respectively, and the PhD degree from the Institute of Automation, Chinese Academy of Sciences (CASIA) in 2004. From 2004 to 2010, he was a research assistant at Imperial College London, United Kingdom, and Monash University, Australia, a research fellow at the University of Melbourne, Australia, and a lecturer at the University of Bath, United Kingdom, respectively. Currently, he is a full professor of the Hundred Talents Program at the National Lab of Pattern Recognition, CASIA. His major research interests include machine learning, pattern recognition, and computer vision. He has widely published in highly ranked international journals such as *IEEE Transactions on Pattern Analysis and Machine Intelligence* and *IEEE Transactions on Information Processing*, and leading international conferences such as CVPR, ICCV, and ICDM. He is a senior member of the IEEE.

Tieniu Tan received the BSc degree from Xi'an Jiaotong University, China, in 1984 and the MSc and PhD degrees from Imperial College of Science, Technology and Medicine, London, in 1986 and 1989, respectively. In 1989, he joined the Computational Vision Group, Department of Computer Science, University of Reading, United Kingdom, where he was a research fellow, a senior research fellow, and a lecturer. In 1998, he joined the National Laboratory of Pattern Recognition (NLPR), Institute of Automation, Chinese Academy of Sciences (CASIA), Beijing, where he is currently a professor and the director of NLPR. He has published more than 300 research papers in refereed journals and conference proceedings in the areas of image processing, computer vision, and pattern recognition. His current research interests include biometrics, image and video understanding, information hiding, and information forensics. He is a fellow of the IEEE and the IAPR.