Sparse Representation With Kernels

Shenghua Gao, Ivor Wai-Hung Tsang, and Liang-Tien Chia

***Abstract*—Recent research has shown the initial success of sparse coding (Sc) in solving many computer vision tasks. Motivated by the fact that kernel trick can capture the nonlinear similarity of features, which helps in finding a sparse representation of nonlinear features, we propose kernel sparse representation (KSR). Essentially, KSR is a sparse coding technique in a high dimensional feature space mapped by an implicit mapping function. We apply KSR to feature coding in image classification, face recognition, and kernel matrix approximation. More specifically, by incorporating KSR into spatial pyramid matching (SPM), we develop KSRSPM, which achieves a good performance for image classification. Moreover, KSR-based feature coding can be shown as a generalization of efficient match kernel and an extension of Sc-based SPM. We further show that our proposed KSR using a histogram intersection kernel (HIK) can be considered a soft assignment extension of HIK-based feature quantization in the feature coding process. Besides feature coding, comparing with sparse coding, KSR can learn more discriminative sparse codes and achieve higher accuracy for face recognition. Moreover, KSR can also be applied to kernel matrix approximation in large scale learning tasks, and it demonstrates its robustness to kernel matrix approximation, especially when a small fraction of the data is used. Extensive experimental results demonstrate promising results of KSR in image classification, face recognition, and kernel matrix approximation. All these applications prove the effectiveness of KSR in computer vision and machine learning tasks.**

***Index Terms*—Face recognition, image classification, kernel matrix approximation, kernel sparse representation, nonlinear mapping, sparse coding.**

I. Introduction

SPARSE coding is a task of reconstructing a given signal by selecting a relatively small subset of basis from a large basis pool, meanwhile keeping the reconstruction error as small as possible. Due to its plausive statistical theory [1], [2], sparse coding is attracting more and more researchers' attention, including those from the computer vision community, and it has also shown its state-of-the-art performance in many computer vision applications, such as image annotation [3], image restoration [4], image classification [5]–[7] *etc*.

There are several reasons accounting for the success of sparse coding in solving computer vision problems. First, the sparseness characteristics ubiquitously exists in many computer vision applications [8], [9]. For example, for face recognition problem [10], the face to be tested can be accurately reconstructed by using a small number of training images with the same category label from the complete training set (sparse reconstruction). For image annotation, the image components can be sparsely reconstructed by using similar components from other relevant images [3], [11], [12]. Such sparseness characteristics is the foundation for the applications of sparse coding.

Secondly, noises commonly exist in images and videos; while sparse coding can automatically select the related basis to reconstruct the feature to be coded, meanwhile can cope with the noise by allowing the reconstruction error. Therefore, sparse coding can be easily applied to feature quantization in Bag-of-Words (BoW) model based image representation [5], [13] or image restoration [4] *etc.*

Existing work based on sparse coding only seeks the sparse representation of the given feature in the original feature space. Recall that kernel trick [14] maps the non-linear features into a higher dimensional feature space, in which features of different categories are linearly separable. In this case, we may find a sparse representation for the features more easily. Therefore, we propose Kernel Sparse Representation (KSR), which is a sparse coding technique in a high dimensional feature space mapped by some implicit mapping function. Another motivation of our KSR comes from the application of Histogram Intersection Kernel (HIK) and sparse coding in feature quantization/coding in BoW image classification framework. On the one hand, recent work [15] shows that HIK based feature quantization which corresponds to the hard assignment in a RKHS can improve the image representation for image classification. On the other hand, sparse coding based soft assignment feature quantization [5] achieves better performance than the hard assignment strategy in the original feature space. Motivated by virtue of these two strategies, we develop KSR using HIK for soft assignment feature coding in the RKHS, which can further enhance the discriminative ability of the image representation and boost the image classification performance.

The core contributions of this paper can be summarized as follows: 1) We propose the KSR framework, which extends sparse coding to a high dimensional feature space. 2) We incorporate KSR into the Spatial Pyramid Matching framework [16] and develop KSRSPM for image classification. KSRSPM can be interpreted as an extension of Sparse Coding based Spatial Pyramid Matching (ScSPM) [5]. We also show that KSR can be interpreted as the sparse version of Efficient Match Kernel (EMK) [17] based image representation. 3) KSR can learn more discriminative sparse codes than sparse coding, and greatly boosts the face recognition performances. 4) KSR

Manuscript received December 27, 2011; revised August 15, 2012; accepted August 16, 2012. Date of publication September 21, 2012; date of current version January 8, 2013. This work was supported by the research grant for the Human Sixth Sense Programme at the Advanced Digital Sciences Center from Singapore's Agency for Science, Technology and Research (A*STAR). The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Wai-Kuen Cham.

S. Gao is with the School of Computer Engineering, Nanyang Technological University, 639798, Singapore, and also with Advanced Digital Sciences Center, 639798, Singapore (e-mail: gaos0004@ntu.edu.sg).

I. W.-H. Tsang and L.-T. Chia are with the School of Computer Engineering, Nanyang Technological University, 639798, Singapore (e-mail: ivortsang@ntu.edu.sg; asltchia@ntu.edu.sg).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TIP.2012.2215620

can also be applied to low-rank kernel matrix approximation problem and achieves good performance especially for the case where the data used for kernel matrix approximation is very small fraction of the whole dataset.

This paper is an extension of our conference paper [18]. Compared with [18], we further improve our work in the following aspects: 1) The implementation issues of optimizing KSR with Feature-Sign Search Algorithm are given in this paper. 2) Motivated by the success of HIK in evaluating the similarity of histogram based features [15], we replace Gaussian kernel with the parameter-free HIK kernel in our KSRSPM framework for feature coding. We first show that KSR using HIK can be interpreted as the soft assignment of HIK based feature quantization [15]. This further enhances the easiness of our KSR framework for feature coding by avoiding additional parameter tuning. Experimental results show that our KSRSPM-HIK usually further improves the image classification accuracy. 3) We perform extensive experiments to compare our KSR with more sparse coding related methods in both image classification and face recognition tasks. 4) We further extend our KSR to solve low-rank matrix approximation in large scale learning tasks, which further proves its effectiveness. 5) We conduct comprehensive analysis for our method, including the effect of different kernels for face recognition and image classification, the effect of different parameters, and experimental settings.

The rest of this paper is organized as follows: In Section II, we briefly review some related work in image classification, face recognition and nonlinear regress with sparse constraint. After that, we will describe the details of KSR, including its objective function and its implementation in Section III. In Section IV, we present KSRSPM and analyze its relationship between ScSPM [5], EMK [17] and HIK based feature quantization in image representation [15]. Then we evaluate its performance on several commonly used image classification datasets. Experimental studies on KSR for the application of face recognition and low-rank kernel matrix approximation will also be presented in Section V and Section VI. Finally, we conclude our work in Section VII.

II. Related Work

In this section, we will review some related work in the following aspects: 1) BoW, SPM, ScSPM and some other sparse coding based image representation models for image classification; 2) Related work of Sparse coding for face recognition; 3) Nonlinear regression with sparse constraint; and 4) Related work of KSR for kernel matrix approximation.

BoW model [19] is widely used in computer vision [15], [20] due to its compact representation and robustness to scale and rotation variance. Generally, it contains three modules: 1) Region selection and representation; 2) Codebook generation and feature quantization/coding; 3) Frequency histogram based image representation (feature pooling). In these three modules, codebook generation and feature quantization are the most important procedures for image representation. The codebook is a collection of primitive patterns used to reconstruct the local features. Each primitive pattern is known as a codeword. Usually k-means is adopted to generate the codebook, and each local feature is quantized to its nearest codeword. However, such hard assignment method may cause severe information loss [21], [22], especially for those features located at the boundary of several codewords. To minimize such information loss, soft assignment [20], [22] was introduced by assigning each feature to more than one codewords. However, the way of choosing parameters, including the weight assigned to the codeword and the number of codewords to be assigned, is not trivial to be determined.

Recently, Yang *et al.* [5] proposed the method of using sparse coding in the codebook generation and feature quantization modules. Sparse coding can learn a better codebook that further minimizes the quantization error than k-means clustering [5]. Meanwhile, the weights assigned to each codeword are learned concurrently. By applying sparse coding to Spatial Pyramid Matching [16] (referred to as: ScSPM), their method achieves promising results in image classification. However, sparse coding based feature coding suffers from the locality information loss. To preserve such locality information, Locality-constrained Linear coding (LLC) [23] and Laplacian sparse coding (LSc) [7] are proposed. Both LLC and LSc achieve better performances for image classification problems. Moreover, Maji *et al.* also propose a feature encoding technique [24] which maps the image representation vector, like SPM, to a new representation, and the classification of such mapped features by using linear SVM has similar or better performance than the classification of original representation with HIK SVM. Such strategy can accelerate the test speed without the loss of accuracy.

Another successful application of sparse coding is face recognition. Face recognition is a classic problem in computer vision, and has a great potential in many real world applications. It generally contains two stages. 1) Feature extraction; and 2) Classifier construction and label prediction. Usually Nearest Neighbor (NN) [25] and Nearest Subspace (NS) [26] are used. However, NN predicts the label of the image to be tested by only using its nearest neighbor in the training data, therefore it can easily be affected by noise. NS approximates the test image by using all the images belonging to the same category, and assigns the image to the category which minimizes the reconstruction error. But NS may not work well in the case where classes are highly correlated to each other [10]. To address these issues, Wright *et al.* proposed a sparse coding based face recognition framework [10], which can automatically select the images in the training set to approximate the test image. Their method achieves excellent performance and exhibits its robust to occlusion, illumination and noise.

Following the work of Wright *et al.*, other sparse coding related face recognition methods are proposed [27]–[29] recently. Collaborative representation for face classification [29] (referred to as CRC) is one representative method among these work. CRC replaces the $\ell 1$ norm with $\ell 2$ norm in sparse coding formulation to solve the face recognition problem. Such representation greatly improves the computational efficiency. Experimental results reported show that the collaborative representation could achieve comparable or better performance in face recognition problem. Meanwhile, experimental results also show that collaborative representation also exhibits its robustness to the weight of $\ell 2$ norm.

Apart from the image classification and face recognition tasks, some attempts [30]–[32] have been made in previous work to solve nonlinear regression with a sparse ($\ell 1$ norm) constraint, such as kernelized LASSO [33], structure modeling with sparse kernels [34] *etc.* Different from these work which only mapped the input (corresponding to the codebook in our formulation) to a RKHS, our KSR formulation maps both the input and output (corresponding to the features to be encoded) to a RKHS. Such a mapping strategy is natural in both feature coding and face recognition applications, which both seek a sparse reconstruction of the features to be encoded by using the codebook entries.

Kernel method has shown great success in solving many real-word problems with non-linear data structures. Given n samples, the corresponding Gram matrix would be an $n \times n$ matrix, which imposes a great burden on both computational cost and space storage/memory, especially for large scale learning tasks. Low-rank matrix approximation provides an effective way for alleviating such memory and computational cost. A number of methods have been proposed for low-rank matrix approximation, such as incomplete Cholesky decomposition [35], [36] Nyström based methods [37] *etc.* In which the Nyström based methods have drawn lots of researchers' attention due to their effectiveness in low-rank matrix approximation. In Nyström low rank approximation, the kernel matrix K is approximated by the production of three low-rank matrix: $K \backsimeq EW^{-1}E'$, in which E ($E \in \mathbb{R}^{n \times m}$) are the columns/rows selected from K, and W are the intersections of the selected rows and columns of K. The sampling scheme is essential for the approximation accuracy. Usually randomized algorithms are adopted [38], [39]. Other than sampling original data, Zhang *et al.* [37] proposed the method of using the results of k-means as the landmark points to construct the E, and they theoretically and experimentally proves their method can further reduce the low-rank matrix approximation error and achieve excellent performance.

III. Kernel Sparse Representation and Its Implementation

In this section, we first describe the notation and preliminaries of sparse coding. Section III-B presents our proposed Kernel Sparse Representation (KSR), and Section III-C gives the implementation details of KSR.

A. Preliminaries of Sparse Coding

Given a feature x ($x \in \mathbb{R}^d$) and a basis pool $U = [u_1, u_2, ..., u_k]$, ($U \in \mathbb{R}^{d \times k}$), sparse coding aims at sparsely and linearly reconstructing the feature to be encoded, namely, $x = v_1 u_1 + v_2 u_2 + \cdots + v_k u_k = Uv$. Herein, *sparseness* means only a small fraction of elements in v are non-zero. The optimization problem of sparse coding can be formulated as follows

$$\min_{v} \quad \|v\|_0 \quad \text{s.t.} \quad x = Uv. \tag{1}$$

Here $\|v\|_0$ means the number of non-zero elements in v. However, the minimization of ℓ_0 norm is an NP hard problem. Donoho proved that “for most large under-determined systems of linear equations, the minimal $\ell 1$-norm near-solution approximates the sparsest near-solution" [1], [2], thus recent research usually formulates the sparse coding problem as the minimization of $\ell 1$ norm. What's more, the reconstruction error also exists in many applications, that is: $x = Uv + \epsilon$. It is hard to determine the reconstruction error in advance, so we can optimize the sparsity of the coefficient and the reconstruction error simultaneously. Thus, the objective of the sparse coding can be rewritten as follows [5], [13], [40]

$$\min_{U,v} \|x - Uv\|_F^2 + \lambda \|v\|_1 \quad \text{s.t.} \|u_m\| \leq 1. \tag{2}$$

The first term of Equation (2) is the reconstruction error, and the second term is used to control the sparsity of the sparse codes v. And λ is the tradeoff parameter used to balance the sparsity and the reconstruction error. Empirically, a larger λ corresponds to a sparser solution.

B. Kernel Sparse Representation

Motivated by nonlinear generalization performance of kernel methods [14], [33] and recent success of Histogram Intersection Kernel (HIK) in computer vision applications [15], [41], we propose a kernel version of sparse representation, namely KSR in short. Essentially, KSR seeks the sparse representation for a mapped feature under the mapped basis in the high dimensional space. Suppose that there exists a kernel $k(\cdot, \cdot)$ induced by feature mapping function ϕ: $\mathcal{R}^d \rightarrow \mathcal{R}^{\mathcal{F}}$ where $d \ll \mathcal{F}$ and $k(u_i, u_j) = \phi(u_i) \cdot \phi(u_j)$ represents a nonlinear similarity between two vectors u_i and u_j, the function maps the input feature and basis to the high dimensional feature space:

$$\begin{gathered} x \xrightarrow{\phi} \phi(x) \\ U = [u_1, u_2, ..., u_k] \xrightarrow{\phi} \mathcal{U} = [\phi(u_1), \phi(u_2), \ldots, \phi(u_k)]. \end{gathered} \tag{3}$$

Then we substitute the mapped features and basis to the formulation of sparse coding, and arrive at the objective of KSR:

$$\min_{U,v} \quad \|\phi(x) - \mathcal{U}v\|_F^2 + \lambda \|v\|_1 \quad \text{s.t. } \kappa(u_i, u_i) \leq 1. \tag{4}$$

Let K_{UU} be a $k \times k$ matrix with $\{K_{UU}\}_{ij} = \kappa(u_i, u_j)$, and $K_U(x)$ be a k-dimensional vector with $\{K_U(x)\}_i = \kappa(u_i, x)$, Equation (4) can be written as:

$$\begin{aligned} \min_{U,v} \quad & \kappa(x, x) + v^T K_{UU} v - 2v^T K_U(x) + \lambda \|v\|_1 \\ \text{s.t.} \quad & \kappa(u_i, u_i) \leq 1. \end{aligned} \tag{5}$$

C. Implementation Issues

The objective of KSR in Equation (5) is the same as that of sparse coding except for the definition of K_{UU} and $K_U(x)$, which can be computed using any kernel. When the linear kernel is used in KSR, K_{UU} and $K_U(x)$ become $U'U$ and $U'x$ respectively, and KSR reduces to standard sparse coding.

```
Algorithm 1 Feature-Sign Search Algorithm for Solving: $\min_{v}\ L(v) + \lambda\|v\|_1$
1: INPUT: The codebook $U$, input feature $x$, and the weight of sparsity term: $\lambda$
   OUTPUT: The kernel sparse codes: $v$.
2: Initialize: $x := \overrightarrow{0}, \theta := \overrightarrow{0}$, active set := {}; where $\theta_i \in \{-1, 0, 1\}$ is the of $sign(v^i)$, which is the sign $i^{th}$ entry of $v$.
3: From zero coefficients of $x$, select $k = \arg max_i |L_v^i|$.
     If $L_v^k > \lambda$, then set $\theta_k := -1$, active set := $\{k\}\cup$ active set.
     If $L_v^k < -\lambda$, then set $\theta_k := 1$, active set := $\{k\}\cup$ active set.
4: Feature-sign step:
     Let $\hat{U}$ and $\hat{L}_{vv}$ be a submatrix of $U$ and $L_{vv}$that contain only the columns corresponding to the active set.
     Let $\hat{v}$, $\hat{K}_U(x)$ and $\hat{\theta}$ be subvectors of $v$, $K_U(x)$ and $\theta$ corresponding to the active set.
     Compute the analytical solution to the resulting unconstrained QP ($min_{\hat{v}}\ L(\hat{v}) + \lambda\hat{\theta}^T\hat{v}$).
       $\hat{v}_{new} = (\hat{L}_{vv})^{-1}(2\hat{K}_U(x) - \lambda\hat{\theta})$
     Perform a discrete line search on the closed line segment from $\hat{v}$ to $\hat{v}_{new}$:
       Check the objective value at $\hat{v}_{new}$ and all points where any coefficient changes sign.
       Update $\hat{v}$ (and the corresponding entries in $v$) to the point with the lowest objective value.
     Remove zero coefficients of from the active set and update $\theta := sign(v)$.
5: Check the optimality conditions:
     (a) Optimality condition for nonzero coefficients:$L_v^j + \lambda sign(v^j) = 0, \forall v^j \neq 0$.
     If condition (a) is not satisfied, go to Step 4 (without any new activation); else check condition (b).
     (b) Optimality condition for zero coefficients:$|L_v^j| < \lambda, \forall v^j = 0$.
     If condition (b) is not satisfied, go to Step 3; otherwise return $v$ as the solution.
```

Given a certain type of kernel function, as U is involved in K_{UU} and $K_U(x)$, thus it is nontrivial to optimize Equation (5). Other than optimizing the codebook U, some recent works show that the codebook initialized with some heuristics can still achieve very good performance [23], [42]. As a compromise, we simply fix U to be a certain codebook generated by some methods when it is not given in its application to feature coding. Moreover, as will be shown in Section IV-D, our KSR can still very satisfactory results. Once the codebook U is given, we can easily extend the Feature-Sign Search Algorithm [13] to optimize the sparse codes.

Denote $L(v) = \kappa(x, x) + v^T K_{UU} v - 2v^T K_U(x)$, then we have the following relationships:

$$L_v = \frac{\partial L(v)}{\partial v} = 2(K_{UU}v - K_U(x))$$
$$L_{vv} = \frac{\partial^2 L(v)}{\partial v^2} = 2K_{UU}. \qquad (6)$$

Subsequently, we can use Algorithm 1 to solve kernel sparse codes. Note, sparse coding, which is actually the case of KSR using linear kernel, is solved by the same algorithm. Thus, the computational cost of KSR is the same as that of sparse coding except for the difference in calculating different kernel matrix. In practice, because the difference in the sparsity of the sparse codes, the computational costs of KSR and sparse coding are a little different.

IV. Application I: KSR for Image Classification

k-means clustering is usually used to generate the codebook in BoW model. In k-means, the whole local feature space $X = [x_1, x_2, \ldots, x_N]$ (where $x_i \in \mathbb{R}^{d\times 1}$) is split into k clusterings $S = [S_1, S_2, \ldots, S_k]$. Denote the corresponding clustering centers as $U = [u_1, u_2, \ldots, u_k] \in \mathbb{R}^{d\times k}$. In hard assignment, each feature is only assigned to its nearest cluster center, and the weight the feature contributing to that center is 1. The objective of k-means can be formulated as the following optimization problem:

$$\min_{U,S}\sum_{i=1}^{k}\sum_{x_j\in S_i}\|x_j - u_i\|^2 = \min_{U,V}\sum_{i=1}^{N}\|x_i - Uv_i\|^2$$
$$\text{s.t. : Card}(v_i) = 1, \|v_i\| = 1, \quad v_i \succeq 0, \quad \forall i. \qquad (7)$$

Here V is a clustering indices, $V = [v_1, v_2, \ldots, v_N]$ (where $v_i \in \mathbb{R}^{k\times 1}$). Each column of V indicates which codeword the local feature should be assigned to.

A. KSR for Feature Coding

To reduce the information loss in feature quantization, the constraint on v_m is relaxed. Meanwhile, to avoid each feature being assigned to too many clusters, the sparse constraint is imposed on v_m. Therefore, given the codebook, the feature coding problem can be formulated as the following sparse coding problem [5]:

$$\min_{U,V}\sum_{i=1}^{N}\|x_i - Uv_i\|^2 + \lambda\|v_i\|_1, \qquad \text{s.t.}\|u_j\| \leq 1, \qquad \forall j. \qquad (8)$$

Following Section III-B, similarly, we can easily extend Equation (8) to KSR for feature coding. In our implementation, we will use HIK in KSR. There are two reasons accounting for this choice of kernel: First of all, HIK is a parameter-free kernel. Secondly, as will be discussed in Section IV-B, KSR using HIK is more suitable for solving SIFT based feature coding problem.

As for the codebook, we simply fix it to be the results of k-means clustering by following the work of Yang *et al.* [23]. Though such codebook may not be optimal, as will be shown in the experiments in Section IV-E, this strategy can still achieve excellent performance for image representation.

B. KSR Using HIK for Feature Coding

SIFT [43] is widely used in image representation due to its good performance for local patch characterization, and it is intrinsically a weighted combination of several histograms of oriented gradients. Traditionally Euclidean distance is used as the distance measurement in codebook generation and feature coding in SIFT feature based BoW model. However, Maji *et al.* [41] pointed out HIK is a more effective measurement for evaluating the similarity/distance for histogram based features than Euclidean distance. Therefore, Euclidean based codebook generation and feature coding are less preferred for image representation. Recently Wu *et al.* proposed a HIK based codebook generation and feature coding strategy (histogram intersection kernel k-means)[1] for image representation [15] and achieved good performance for image classification. HIK based codebook generation and feature coding can be formulated as the following objective function (We use the same notations with previous sections):

$$\min_{U,V} \sum_{i=1}^{N} \|\phi(x_i) - Uv_i\|^2 \quad \text{s.t.} \mathrm{Card}(v_i) = 1, \\ \|v_i\| = 1, \quad v_i \geq 0, \quad \forall i. \tag{9}$$

Here ϕ is a nonlinear mapping, and $\kappa_{HI}(x, y) = \phi(x) \cdot \phi(y) = \sum_i \min(x_i, y_i)$.

Such HIK based feature coding is essentially the hard assignment feature coding strategy in the space mapped by ϕ. As mentioned by Gemert *et al.* [22], hard assignment will cause the information loss for the original features. Therefore, in this section, we introduce the soft assignment feature coding, and it is nature for us to impose the sparse constraints on the cluster index because of the good performance of sparse coding for feature coding in original feature space [5]. Thus we formulate the modified HIK based feature coding problem as follows

$$\min_{U,V} \sum_{i=1}^{N} \|\phi(x_i) - Uv_i\|^2 + \lambda\|v_i\|_1 \quad \text{s.t.} \|u_j\| \leq 1, \quad \forall j \tag{10}$$

Therefore, HIK based KSR is a soft sparse feature coding version of HIK based feature coding. Because of the good performance of HIK based feature coding and sparse coding for image representation, it can be expected that KSR using HIK for feature coding can further improve the image representation ability.

C. Connection Between KSR and EMK

EMK [17] is an efficient feature approximation technique for those features mapped to a higher dimensional feature space. Compared with image classification with linear SVM for images represented by BoW model, EMK can better characterize the similarity among different images, and improve the image classification accuracy.

Let $X = [x_1, x_2, \ldots, x_p]$ be a set of local features in one image, and $V(x) = [v_1(x), v_2(x), \ldots, v_p(x)]$ are the corresponding clustering index vector in Equation (7). In BoW model, each image is presented by a normalized histogram $\bar{v}(X) = \frac{1}{|X|}\sum_{x \in X} v(x)$, which characterizes its codeword distribution. By using linear classifier, the resulting similarity between different images is calculated by:

$$\begin{aligned} K_B(X, Y) &= \frac{1}{|X||Y|} \sum_{x \in X, y \in Y} v(x)^T v(y) \\ &= \frac{1}{|X||Y|} \sum_{x \in X, y \in Y} \delta(x, y) \end{aligned} \tag{11}$$

where $\delta(x, y) = \begin{cases} 1, & v(x) = v(y) \\ 0, & \text{otherwise.} \end{cases}$ $\delta(x, y)$ is positive definite kernel, which is used to measure the similarity between two local features. However, such hard assignment based local feature similarity measurement causes the information loss, and consequently reduces classification accuracy. Thus, a continuous kernel is introduced to more accurately measure the similarity between local feature x and y:

$$K_S(X, Y) = \frac{1}{|X||Y|} \sum_{x \in X, y \in Y} k(x, y). \tag{12}$$

Here $k(x, y)$ is a positive definite kernel, which is referred to as local kernel. This is related to the normalized sum match kernel [44], [45].

Due to the large amount of local features, directly using local kernel is both storage and computation prohibitive for image classification. To decrease the computation cost, EMK is introduced. Under the definition of finite dimensional kernel function [17], $k(x, y) = \phi(x)^T \phi(y)$, $\phi(x)$ can be approximated by using low dimensional features $\bar{v}_x$ in the space spanned by k basis vectors $H = [\phi(u_1), \phi(u_2), \ldots, \phi(u_k)]$:

$$\min_{H, v_x} \|\phi(x) - Hv_x\|^2. \tag{13}$$

In this way, each image can be represented by $\bar{v}(X)_{new} = \frac{1}{|X|} H \sum_{x \in X} v_x$ beforehand. As a result, the computational speed can be accelerated.

In summary, EMK is consisted of two stages: i) $x \xrightarrow{\phi} \phi(x)$: Map the feature to new feature space; ii) $\phi(x) \xrightarrow{H} \bar{v}_x$: Reconstruct/approximate $\phi(x)$ by using the basis H. Therefore, we may say that the improvement of EMK in images classification accuracy actually results from the usage feature mapping, which evaluates the similarity between local features more accurately and consequently boosts the image representation.

In EMK, it uses the approximated feature Hv_x which corresponds to the approximation of mapped feature ($\phi(x)$) for image representation. In other words, it directly uses the features in a RKHS for image representation. However, directly using original feature for image classification may cause overfitting for many learning-based classifiers [21]. Note that the objective of Equation (13) is similar to the objective of feature coding in BoW model, therefore it can also be seen as a feature coding formulation for features mapped to a high feature dimensional space. To avoid overfitting,

[1] Besides HIK based feature coding, Wu *et al.* also proposed a one class SVM method which can further boost the feature coding performance. But such one class SVM (ocSVM) still belongs to hard assignment feature coding in a RKHS.

TABLE I

PERFORMANCE COMPARISONS ON UIUC-SPORT DATASET AND COREL 10 DATASET(%)

	UIUC-Sport	Corel 10
HIK+ocSVM [15]	83.54±1.13	NA
Spatial Markov Model [52]	NA	77.9
AC (400) [24]	80.36±0.57	84.90±1.33
AC (1024) [24]	81.53±0.81	84.47±0.97
EMK [17]	84.55±1.21	88.80±1.79
ScSPM [5]	82.74±1.46	86.2±1.01
LLC (1024) [23]	83.09±1.30	87.93±1.04
LLC (4096) [23]	85.36±1.02	88.57±1.27
LScSPM	85.31±0.51	88.4±0.78
KSRSPM-Gaussian [18]	84.92±0.78	89.43±1.27
KSRSPM-HIK	**86.85±0.45**	**90.50±1.07**

and following the idea of BoW model, we propose to use v_x for image classification. Because of the good property of sparse constraint for feature coding [5], we also add the sparse constraint on v_x in the objective of EMK. Then we arrive at the following formulation

$$\min_{H,v_x} \quad \|\phi(x) - Hv_x\|^2 + \lambda\|v_x\|_1. \tag{14}$$

By comparing the objectives of KSR and the above modified EMK, we may say that KSR is the sparse EMK. Because of the good performance of EMK, it is expected that KSR based feature coding can further improve the image representation and the image classification accuracy.

D. Feature Pooling and Image Representation

Following the work of [5], [46], we use maximum pooling method to represent the images. Maximum pooling uses the largest responses to each basic pattern to represent the region. More specifically, suppose one image region has D local features, and the codebook size is k. After maximum pooling, each image will be represented by a k dimensional vector y, and the l^{th} entry is the largest response to the l^{th} basis vector of all the sparse codes in the selected region (v_D is the sparse codes of the D^{th} feature in this local region, and v_{Dl} is the l^{th} entry of v_D):

$$y_l = \max\{|v_{1l}|, |v_{2l}|, \ldots, |v_{Dl}|\}. \tag{15}$$

To preserve the spatial information, SPM model [16] is used. SPM divides the image into increasingly finer regions. In our implementation, we use the top three layers, i.e., we divide each image into 1 × 1, 2 × 2, 4 × 4 subregions in each layer respectively. Then we perform maximum pooling on sparse codes obtained by HIK based KSR for the representation of each subregion. Finally the representations in each layer are weighted and concatenated to represent the image. Following the work of Yang *et al.* [5], we set equal weights to all layers.

E. Experiments

1) Experimental Setup: To be consistent with previous work [5], [16], [17], we also use SIFT feature for local patch characterization. Specifically, following the work of [23], we use dense grid sampling strategy, and fix the step size to be 8, and extract SIFT at three different patch size scales: 16, 25, and 31. We also resize the maximum side (width/length) of each image to 300 pixels. After obtaining the SIFT, we use $\ell 2$-norm to normalize the feature length to 1. $(5.0 \sim 8.0) \times 10^4$ features are randomly selected to generate codebook for each dataset. For the codebook size, we set $k = 4096$ [23]. We simply fix $\lambda = 0.20$ for all the datasets. We use liblinear SVM [47] due to its advantage in speed and excellent performance in sparse coding based image classification [23][2]. All the results reported for each dataset are based on six independent experiments in which the training/test images are randomly split.

2) Dataset Descriptions: We evaluate our method on three commonly used datasets in image classification. 1) **UIUC-Sport** [48]. It contains 1792 sports images, which belong to 8 different categories, and the number of images ranges from 137 to 250 per category. Following the work of Wu *et al.* [15], we randomly select 70 images from each category as training data, and randomly select another 60 images from each category as test data. 2) **Corel 10** [49]. It contains 10 categories, and each category contains 100 images. Following the work of Lu *et al.* [49], we randomly select 50 images as training data and use the rest as test data. 3) **Caltech 256** [3]. It is a very challenging dataset in both image content and dataset scale. First of all, compared with Caltech 101, the objects in Caltech 256 contains larger intra-class variance, and the object locations are no longer in the center of the image. Second, Caltech 256 contains 29780 images, which are divided into 256 categories. More categories will inevitably increase the inter-class similarity, and increase the performance degradation. We evaluate the method under four different settings: selecting 15, 30, 45, 60 per category as training data respectively, and use the rest as test data. The results of different methods on UIUC-Sport and Corel 10 are listed in Table I, and the results of different methods on Caltech 256 are listed in Table II.

3) Results Analysis: From Table I and Table II, we can see that our KSRSPM outperforms EMK, ScSPM, HIK based feature coding (HIK+ocSVM [15]) and LLC on UIUC-Sport, Caltech 256 and Corel 10 datasets. We can also see that KSRSPM-HIK usually achieves better performance than KSRSPM-Gaussian. Moreover, t-test[4] shows that the improvement of KSRSPM-HIK over ScSPM is considered be to extremely statistically significant by conventional criteria. As for the computational cost, experimentally our HIK based KSR costs about $(5.5 \pm 3.2) \times 10^{-3}$ seconds to encode a SIFT feature, and sparse coding costs about $(4.4 \pm 3.8) \times 10^{-3}$

[2] The performance of baseline methods, like HIK+ocSVM, SPM, ScSPM, LScSPM, KSRSPM-Gaussian, *etc.*, which also use SIFT feature, are directly copied from the related paper or implemented by following the same setting of related papers. The results of EMK reported are based its implementation downloaded from the authors' website. For Additive Classifier (AC), we use the SPM for image representation (Following the work of SPM, we set $k = 400$ in k-means. For fair comparison, following the work of ScSPM we also test the case when $k = 1024$ on UIUC-Sport and Corel 10. Due to computational complexity, we only test the case of $k = 400$ on Caltech 256.). Because different feature is used in LLC, we therefore implement LLC by keeping the same experimental setting with our KSRSPM-HIK.

[3] Available at www.vision.caltech.edu/Image_Datasets/Caltech256/.

[4] Available at http://www.graphpad.com/quickcalcs/ttest1.cfm?Format=SD.

TABLE II
PERFORMANCE COMPARISON ON CALTECH 256 DATASET (%)
(KC: KERNEL CODEBOOK)

ImgNo	15	30	45	60
SPM [50]	NA	34.10	NA	NA
AC(400) [24]	22.80±0.06	27.85±0.20	30.68±0.32	32.48±0.28
KC [22]	NA	27.17±0.46	NA	NA
ScSPM [5]	27.73±0.51	34.02±0.35	37.46±0.55	40.14±0.91
LLC (1024)	27.74±0.32	32.07±0.24	35.09±0.44	37.79±0.42
LLC (4096)	32.68±0.27	39.62±0.15	43.85±0.23	46.11±0.27
LScSPM [7]	30.00±0.14	35.74±0.10	38.54±0.36	40.43±0.38
KSRSPM-Gaussian [18]	29.77±0.14	35.67±0.10	38.61±0.19	40.30±0.22
KSRSPM-HIK	**33.61±0.34**	**40.63±0.22**	**44.41±0.12**	**47.03±0.35**

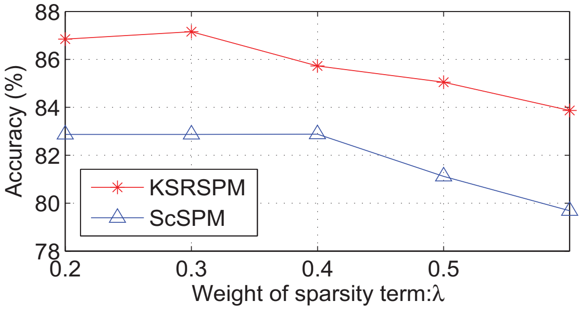

Fig. 1. Relationship between the weight of sparsity term (λ) and image classification accuracy on UIUC-Sport dataset.

seconds to encode a SIFT feature (Both use the feature sign search algorithm and the same codebook). Need to mention that in original ScSPM [5], the codebook size is 1024, and as indicated in LLC [23] that "using a larger codebook helps the performance." So following the work of LLC, we also fix the codebook size to be 4096. By using the same features, sparse coding costs about 3.2 seconds to encode each image (codebook size is 1024), and our KSRSPM costs about 17.4 seconds to encode each image (codebook size is 4096) on Corel 10 dataset. But we can see our KSRSPM improves the performance a lot compared with ScSPM.

4) Parameter Selection: The weight of sparsity term λ is an important factor in KSRSPM. To ease the parameter selection, we experimentally test its effect on the performance of image classification on UIUC-Sport dataset. We vary λ from 0.2 to 0.6 and plot the corresponding image classification accuracy in Fig. 1. We can see that KSRSPM achieves good performance when λ is fixed to be 0.2 or 0.3. In our experiments, we simply fix $\lambda = 0.2$. We also compute the accuracy of ScSPM under the same weight of sparsity and also plot it in Fig. 1. We can see that our KSRSPM can consistently outperform ScSPM and achieve good performance with a large range of λ.

V. APPLICATION II: KSR FOR FACE RECOGNITION

Another application of sparse coding is face recognition [10]. In this section, we first reformulate the objective of sparse coding for face recognition, and then extend it to KSR.

A. Sparse Coding for Face Recognition

For face recognition, "If sufficient training samples are available from each class, it would be possible to represent the test samples as a linear combination of those training samples from the same class [10]."

Suppose there are N classes in all, and the training instances in the i^{th} class are $A_i = [a_{i,1}, \ldots, a_{i,n_i}] \in \mathbb{R}^{d \times n_i}$, in which each column corresponds to one instance. Let $A = [A_1, \ldots, A_N] \in \mathbb{R}^{d \times \sum_{i=1}^{N} n_i}$ be the training set, and $y \in \mathbb{R}^{d \times 1}$ be the test sample. When small dense noise exists, the problem for face recognition can be formulated as follows [10]

$$\min_x \|x\|_1 \quad \text{s.t.} \|y - Ax^T\|_2 \leq \epsilon \tag{16}$$

Sparse coding based image recognition aims at selecting only a few images from all the training instances to reconstruct the images to be tested. Let $\alpha_i = [\alpha_{i,1}, \ldots, \alpha_{i,n_i}] (1 \leq i \leq N)$ be the coefficients corresponds to A_i in x. The reconstruction error by using the instances from the i^{th} class can be computed as: $r_i(y) = \|y - A_i \alpha_i\|_2$. Then the test image is assigned to the category that minimizes the reconstruction error:

$$\text{class label of } y := \arg \min_i \{r_1(y), \ldots, r_N(y)\} \tag{17}$$

B. KSR for Face Recognition

Kernel method can ensure that the features belonging to the same category are closer to each other [14]. Thus we apply KSR to face recognition. Other than estimating the noisy (reconstruction error) level (ϵ), we add the reconstruction error to the objective by introducing a Lagrange Multiplier (λ) [29]. By mapping features to a high dimensional space: $y \to \phi(y)$, $A = [a_{1,1}, \ldots, a_{N,n_N}] \to \mathcal{A} = [\phi(a_{1,1}), \ldots, \phi(a_{N,n_N})]$, we arrive at the objective of KSR for face recognition:

$$\min_x \quad \lambda\|x\|_1 + \|\phi(y) - \mathcal{A}x\|_2^2 \tag{18}$$

In which the parameter λ is used to balance the weight between the sparsity and the reconstruction error. Following the work of Wright *et al.*, the test image is assigned to the category which minimizes the reconstruction error in the high dimensional feature space.

C. Performance Evaluation

1) Experimental Setup: We evaluate our method on Extended Yale B dataset [51] and AR dataset [53]. Extended Yale B contains 38 categories, and 2414 frontal-face images. The cropped image size is 192 × 168. Following the work of [10], we randomly select a half of images in each category as training images, and use the rest as test images. AR dataset contains over 4000 frontal face images corresponding to 126 persons, and the images include more facial variations, including facial expression, illumination and occlusions (sun glasses and scarf). Thus AR is more challenge than Extended Yale B. We choose 47 male persons and 43 female persons for our evaluation. For each persons, 26 images are taken in two separate sessions: in which 14 images only contain variance in illumination and facial expression. Following the work of

TABLE III
EFFECT OF SPARSITY PARAMETER: 56D DOWNSAMPLE FEATURE (HERE SPARSITY IS PERCENTAGE OF NONZERO ELEMENTS IN SPARSE CODES)

λ	10^{-1}	10^{-2}	10^{-3}	10^{-4}	10^{-5}	10^{-6}	10^{-7}	0
Sparsity (%)	0.58	0.75	0.88	2.13	4.66	8.35	16.69	-
Reconstruction Error	0.2399	0.1763	0.1651	0.1113	0.0893	0.0671	0.0462	-
Time (sec)	0.0270	0.0280	0.0299	0.0477	0.2445	0.9926	6.2990	0.00066
Accuracy (%)	76.92	84.12	85.19	90.32	91.65	93.30	93.47	84.37

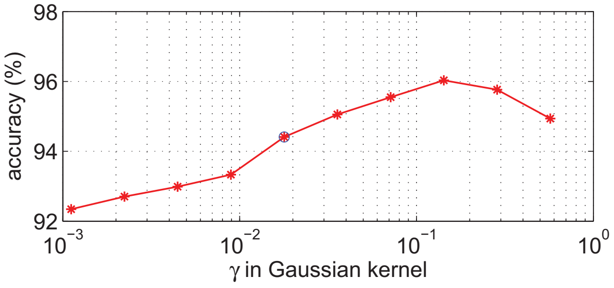

Fig. 2. Test accuracies of kernel parameter for face recognition. The blue circle indicates the parameter we used.

TABLE IV
PERFORMANCES (%) OF DIFFERENT KERNELS ON EXTENDED YALE B DATASET (FEATURE DIMENSION: 30D)

	Eigen	Laplacian	Down-Sampling	Fisher
Gaussian	89.01	88.86	83.57	88.93
Polynomial	**91.74**	**92.06**	**87.20**	**91.54**
IDK	70.33	80.35	61.65	87.12
ISDK	85.90	88.20	80.48	90.71
eHIK	80.34	82.77	51.18	85.70
HIK	NA	NA	48.91	NA

[10], we use 7 of the 14 images as training images and use the rest as test images. We use the following features for comparison: LaplacianFace [54], EigenFace [55], FisherFace [56] and Downsample feature [10], and each feature is normalized to unit length by using $\ell 2$ norm.

2) Parameter Evaluation: **Effect of $\boldsymbol{\lambda}$:** Since the weight of the sparsity term (λ) in KSR affects face recognition performance, we first evaluate the effect of λ by using 56D Downsample feature in Extended Yale B dataset. Here we use Gaussian kernel: $\kappa(x_1, x_2) = exp(-\gamma \|x_1 - x_2\|^2)$ and fix its parameter $\gamma = 1/d$, d is the feature dimension. We list the results based on different λ in Table III. When $\lambda \neq 0$ and λ is in the range we observed, as λ decreases, the performance increases, and the proportion of non-zero elements in coefficients also increases, but the computational time increases. When $\lambda = 0$, the performance is not as good as that in the case of $\lambda \neq 0$. Thus, this shows the effectiveness of the sparsity term.

Effect of $\boldsymbol{\gamma}$: When Gaussian kernel is used in KSR, the kernel parameter γ is also very important and affects the face recognition accuracy. To depict the relationship between γ and face recognition accuracy, we set $\gamma = 2^n/d$. Here d is the feature dimension and n increases from -4 to 5 with step size 1. We plot the relationship between γ and face recognition accuracy in Fig. 2. In our experiments, we use 56D Downsample Feature on Extended Yale B dataset, and the weight of sparsity term $\lambda = 10^{-5}$. Fig. 2 shows that KSR can achieve good performance in a wide range of γ values. For simplification, in the following experiments, we set $\gamma = 1/d$.

3) Effect of Kernel Selection for KSR: We also evaluate the effect of different kernels for KSR in face recognition on Extended Yale B dataset (feature dimension: 30D): Gaussian Kernel, Polynomial Kernel: $(1 + x^T y)^b$, Inverse Distance Kernel(IDK): $\frac{1}{1+b\|x-y\|}$, Inverse Square Distance Kernel(ISDK): $\frac{1}{1+b\|x-y\|^2}$, HIK[5] and Exponential HIK (eHIK): $\sum_i min(e^{bx_i}, e^{by_i})$[6].

Table IV shows that different kernels have different impact on different features. Polynomial kernel and Gaussian kernel usually achieves good performance for all types of features. Therefore, Gaussian kernel and Polynomial kernel will be used in our comprehensive studies. We also notice that both HIK and eHIK achieve poor face recognition accuracy especially for downsampling feature, and this may result from the fact that HIK is usually used to characterize the similarity between two histograms, and it is not suitable for downsampling feature.

4) Result Comparison: We test the performance of KSR-Gaussian and KSR-Polynomial on Extended Yale B and AR datasets. Considering both the computational cost and the accuracy in Table III, we set $\lambda = 10^{-5}$ for KSR-Gaussian, and $\lambda = 10^{-3}$ for KSR-Polynomial. The experimental results are listed in Table V. All the results are based on 10 times independent experiments. Experimental results show that KSR can outperform both sparse coding and CRC[7] in face recognition. Especially for low dimensional features, the improvement is significant, for example, KSR-Gaussian outperforms Sc by 13.65% (60.48% vs. 74.13%) on AR dataset for downsampling feature (28D). Moreover, we also note that KSR-Polynomial usually outperforms KSR-Gaussian on Extended Yale B dataset, but KSR-Gaussian usually outperforms KSR-Polynomial on AR dataset.

5) Similarity Between the Sparse Codes for KSR: To further illustrate the performance of KSR, we calculate the similarity between the sparse codes of KSR and Sc for the instances

[5] Due to negative values in EigenFace, FiserFace and LaplacianFace features, HIK is not applicable to those features.

[6] For the concern of efficiency, we set $\lambda = 10^{-5}$ in Gaussian Kernel, and $\lambda = 10^{-3}$ in other kernels. For Polynomial Kernel, b = 3; For IDK, ISDK and eHIK, b = 1.

[7] For CRC, we also set the weight of $\ell 2$ term to 10^{-5}. According to [29], CRC achieves good performance under such setting.

TABLE V
PERFORMANCE OF DIFFERENT METHODS FOR FACE RECOGNITION ON EXTENDED YALE B DATASET AND AR DATASET (%)

		Extended Yale B				AR			
	Dimension	30D	56D	120D	504D	28D	54D	126D	512D
Eigen	CRC [29]	67.97	85.62	94.41	98.55	52.06	76.67	87.30	87.62
	Sc [10]	86.5	91.63	93.95	96.77	70.32	80.32	83.81	89.50
	KSR-Polynomial	**91.74**	**95.75**	97.42	98.28	77.78	85.08	87.62	89.68
	KSR-Gaussian	89.01	94.42	**97.49**	**99.16**	**78.89**	**85.34**	**89.05**	**90.48**
Laplacian	CRC [29]	72.25	86.89	94.54	97.91	54.92	80.32	87.94	83.33
	Sc [10]	87.49	91.72	93.95	96.52	67.46	81.11	84.60	84.29
	KSR-Polynomial	**92.25**	**95.14**	97.07	**99.11**	76.67	85.08	87.30	87.14
	KSR-Gaussian	88.86	94.24	**97.11**	98.12	**77.78**	**85.71**	**87.78**	**89.10**
Downsample	CRC [29]	67.97	82.84	92.93	97.23	44.92	69.21	83.17	71.90
	Sc [10]	74.57	86.16	92.13	97.1	60.48	75.56	86.51	88.73
	KSR-Polynomial	**87.20**	**93.09**	**95.81**	97.47	73.81	81.75	86.19	88.89
	KSR-Gaussian	83.57	91.65	95.31	**97.80**	**74.13**	**82.70**	**87.14**	**90.32**
Fisher	CRC [29]	68.10	NA	NA	NA	54.60	76.03	NA	NA
	Sc [10]	86.91	NA	NA	NA	66.98	81.27	NA	NA
	KSR-Polynomial	**91.74**	NA	NA	NA	78.25	85.56	NA	NA
	KSR-Gaussian	88.93	NA	NA	NA	**79.37**	**85.87**	NA	NA

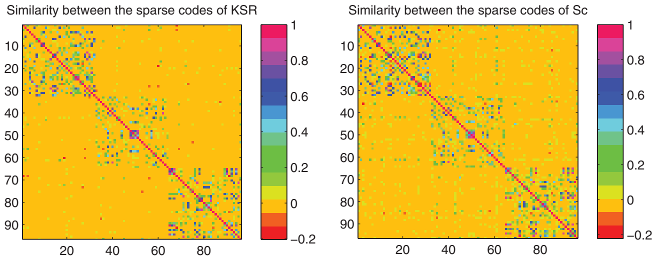

Fig. 3. Similarity between the sparse codes of KSR and Sc.

from three classes (each class contains 32 images) in Extended Yale B dataset. We list the results in Fig. 3, in which the entry located at (i, j) is the sparse codes similarity measured by normalized correlation between image i and j. We know that a good sparse coding method can make the sparse codes belonging to the same class more similar, therefore, the sparse codes similarity should be block-wise. From Fig. 3 we can see that our KSR can get more discriminative sparse codes than sparse coding, which facilitates the better performance of in our face recognition application.

VI. APPLICATION III: KSR FOR LOW-RANK MATRIX APPROXIMATION

As mentioned aforehand, low-rank matrix is an important technique for solving large scale learning tasks. It greatly alleviates the computational cost and memory in the computation process. In this section, we will show how KSR can be used for low-rank matrix approximation.

A. KSR for Low-Rank Matrix Approximation

Consider the first term in the objective of KSR: $\|\phi(x) - \mathcal{U}v_x\|^2$. Given the codebook features in the high dimension space $\mathcal{U} = [\phi(u_1), \phi(u_2), \ldots, \phi(u_k)]$, KSR automatically selects some entries from the codebook to linearly approximate the feature $\phi(x)$: $\phi(x) \simeq \mathcal{U}v_x$. Based on such equation, we can approximate the kernel $\kappa(x, y)$ as follows

$$\kappa(x, y) = \phi(x)^T\phi(y) \simeq (\mathcal{U}v_x)^T(\mathcal{U}v_y) = v_x K_{UU} v_y \quad (19)$$

Here K_{UU} is also the kernel matrix of the codebook features in the original space. Denote the training set as X, which contains n instances in all. The Gram matrix corresponds to X is denoted as K $(K \in \mathbb{R}^{n\times n})$. Following the Equation (19), we can approximate the Gram matrix by using the results of KSR in the following way

$$K \simeq V'K_{UU}V = (V'K^{\frac{1}{2}})(V'K^{\frac{1}{2}})' \quad (20)$$

Here V $(V \in \mathbb{R}^{k\times n})$ is the sparse codes matrix of the training data. In a large scale problem, the number of features is much larger than the size of the codebook: $n \gg k$. Equations (20) illustrates that given the training data and codebook, we can use the sparse codes and kernel matrix of the codebook features to approximate the kernel, and such approximation is a kind of low-rank matrix approximation. Similarly to Nyström methods, the size of the codebook k can be determined based on the scale of the problem and the accuracy requirement. Larger k corresponds to more accurate approximation, which, however, will increase the computational cost.

There are some advantages of using KSR for low-rank matrix approximation. Firstly, KSR uses the sparse codes for low-rank matrix approximation. The sparse structure thus can further decrease the computational cost and storage/memory required compared with Nyström based methods [37], [57]. Secondly, KSR for low-rank matrix approximation is essentially the linear kernel computation in the mapped high dimensional feature space. By using the KSR to encode both the training and test data, it can reduce the variance/noise within the data distribution by adding the sparsity term in Equation (4). What's more, the performance of KSR for low-rank matrix approximation is quite robust. Even if the codebook size is very small, KSR can still achieve very good performance, which will be demonstrated in the experiments.

B. KSR for Least Square SVM (LS-SVM)

Denote the kernel matrix and the training labels corresponding to the kernel matrix are K and y ($y \in \{\pm 1\}^{n\times 1}$). The LS-SVM classifier [58] can be written as $f(x) = \sum_{i=1}^{n} \alpha_i y_i \kappa(x, x_i)$, where $b = y'M^{-1}\mathbb{1}/y'M^{-1}y$, $\alpha = M^{-1}(\mathbb{1} - by)$, $Y = diag(y)$, $M = Y(K+I/C)Y$, here $C > 0$ is regularization parameter, and $\mathbb{1}$ is a vector whose entries are all 1.

We can easily get that $M^{-1} = Y(K + I/C)^{-1}Y$. The computational cost for the calculation of $(K + I/C)^{-1}$ is $O(n^3)$ and memory cost for such matrix is $O(n^2)$. For large scale problem, such cost is very high. For example, if the training number is 40000, then the kernel matrix will result in a "out of the memory" problem (16G RAM). According to the Woodbury formula [57], given the low-rank matrix approximation $K \simeq GG'(G \in \mathbb{R}^{n\times k})$, the following equation holds ($\delta > 0$)

$$(K + \delta I)^{-1} \simeq \frac{1}{\delta}(I - G(\delta I + G'G)^{-1}G') \tag{21}$$

In KSR, $G = V'K^{\frac{1}{2}}$. By using low-rank matrix approximation and Woodbury formula, the computational cost and memory can be greatly reduced to $O(k^2 n)$ and $O(kn)$ respectively. It would be made feasible for such calculation by some properly selected k.

C. Experiments

1) Experimental Setup: To evaluate the performance of KSR for low-rank matrix approximation, we use it for solving LS-SVM problem on USPS dataset[8] and MNIST dataset[9]. USPS and MNIST are two datasets for digit number recognition. Each dataset contains 10 classes (digit 0-9). USPS contains 7 291 training images and 2007 test images, and the feature dimension is 256. MNIST contains 60 000 training images and 10 000 test images, and feature dimension is 780. To speed up the sparse coding, we reduce the feature dimension of MNIST dataset to 256D by using PCA, which can still preserve about 98% energy. We use one-vs.-all strategy to train the LS-SVM classifier. The regularization parameter C in LS-SVM is set to be 1.

2) Performance Comparison: In this part, we compare our method with the improved Nyström method [37], which is proved theoretically and experimentally to be able to achieve better performance than general Nyström method and other low-rank matrix approximation methods, such as SVD, ICD, Drineas's method [38] *etc*. For fair comparison with improved Nyström method, we also use the results of k-means clustering as the codebook U. For the kernel parameter γ in Gaussian kernel, we set it as the average squared distance between the training data points and the mean of all training instances [37], [59]. To overcome the variance caused by k-means clustering, we repeat the experiment 10 times. To avoid the memory cost problem in standard LS-SVM, We use the first 10,000 instances in training set as training data in MNIST dataset in the experiments of this section. The accuracy on the two datasets are given in Fig. 4.

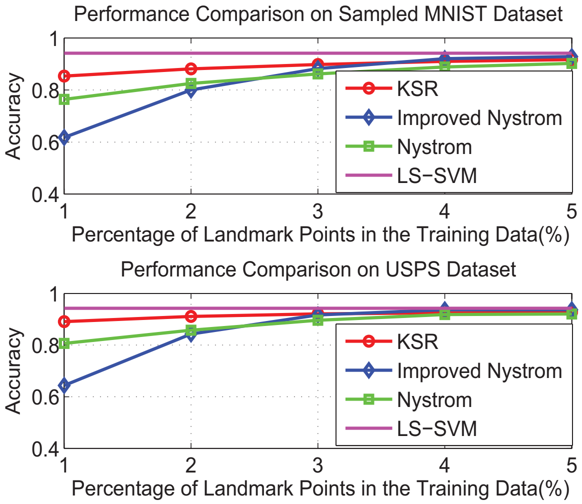

Fig. 4. Accuracy of different methods on MNIST and USPS datasets.

Results show that KSR achieves very good performance for low-rank matrix approximation. Please note that our KSR is very robust to the size of codebook features. Even if the codebook size is only a small fraction (e.g., 1%-3%) of the whole training data where the Nyström method and improved Nyström method are not satisfactory, KSR still achieves relative good accuracy. This may hint that our KSR may especially useful for the case where data used for matrix approximation is very small of the fraction of the whole data. As the data number used for low-rank matrix approximation (k) increases, the accuracy increases simultaneously. When more data are used as codebook features, the performance of KSR is comparable to that of improved Nyström method, and gradually approaching to that of LS-SVM without approximation.[10]

3) Computational Cost: To illustrate the computational cost difference between our method and improved Nyström method [37], we use all the 60 000 training data of MNIST and one-vs.-all strategy to train the LS-SVM classifier. The test bed is Matlab 2008b installed on Windows server with 64bit OS with 16G RAM. We also project the data to the 256D feature space by using PCA. Due to the large kernel matrix (60,000×60,000), which will cause "out of memory" problem, it is impossible to directly use LS-SVM directly. We have to resort to low-rank matrix approximation technique. In experiment, we set the size of the codebook to 200. We repeat the experiment 10 times. In terms of computational time, improved Nyström method costs 11.49 hours while KSR only costs 3.81 hours. This is because the sparse matrix can accelerate the computation.

[8] Available at http://www-i6.informatik.rwth-aachen.de/keysers/usps.html.

[9] Available at http://www.csie.ntu.edu.tw/cjlin/libsvmtools/datasets/multiclass.html#mnist.

[10] It is worthy noting that recently some more advanced and more complicated classification methods can achieve even better performance on these two datasets, but here we only use the LS-SVM, and we want to show that effectiveness of of KSR in low-rank matrix approximation in terms of its simplicity and its good performance in the case of using a even smaller part of data for kernel matrix approximation.

VII. Conclusion

In this paper, we propose a new technique: Kernel Sparse Representation, which is the sparse coding technique in a high dimensional feature space mapped by some implicit feature mapping. We apply Kernel Sparse Representation to image classification, face recognition, and low-rank kernel matrix approximation. For image classification, we show its relationship to the previous work: Sparse coding [5], EMK [17] and HIK based feature quantization [15]. For face recognition, Kernel Sparse Representation can learn more discriminative sparse codes for face category identification. Experimental results on several publicly available datasets show that our Kernel Sparse Representation achieves good performance for both image classification and face recognition. Moreover, KSR also finds good kernel matrix approximation to speed up training processing and achieve good recognition performances on two digital recognition datasets. These results prove the effectiveness of our Kernel Sparse Representation.

Acknowledgment

The work is mainly done when S. Gao was in NTU, Singapore.

References

[1] D. Donoho, "For most large underdetermined systems of linear equations, the minimal ℓ_1-norm solution is also the sparsest solution," *Commun. Pure Appl. Math.*, vol. 59, no. 6, pp. 797–829, Jun. 2006.

[2] D. Donoho, "For most large underdetermined systems of linear equations, the minimal ℓ_1-norm near-solution approximates the sparsest near-solution," *Commun. Pure Appl. Math.*, vol. 59, no. 7, pp. 907–934, Jul. 2006.

[3] C. Wang, S. Yan, L. Zhang, and H.-J. Zhang, "Multi-label sparse coding for automatic image annotation," in *Proc. Comput. Vis. Pattern Recognit.*, 2009, pp. 1643–1650.

[4] J. Mairal, F. Bach, J. Ponce, G. Sapiro, and A. Zisserman, "Non-local sparse models for image restoration," in *Proc. Int. Conf. Comput. Vis.*, 2009, pp. 2272–2279.

[5] J. Yang, K. Yu, Y. Gong, and T. Huang, "Linear spatial pyramid matching using sparse coding for image classification," in *Proc. Comput. Vis. Pattern Recognit.*, 2009, pp. 1794–1801.

[6] J. Mairal, F. Bach, J. Ponce, G. Sapiro, and A. Zisserman, "Discriminative learned dictionaries for local image analysis," in *Proc. Comput. Vis. Pattern Recognit.*, 2008, pp. 1–8.

[7] S. Gao, I. W. Tsang, L.-T. Chia, and P. Zhao, "Local features are not lonely–Laplacian sparse coding for image classification," in *Proc. Comput. Vis. Pattern Recognit.*, 2010, pp. 3555–3561.

[8] B. Cheng, J. Yang, S. Yan, and T. Huang, "Learning with ℓ^1 graph for image analysis," *IEEE Trans. Image Process.*, vol. 19, no. 4, pp. 858–866, Apr. 2010.

[9] M. Wang, X.-S. Hua, R. Hong, J. Tang, G.-J. Qi, and Y. Song, "Unified video annotation via multigraph learning," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 19, no. 5, pp. 733–746, May 2009.

[10] J. Wright, A. Y. Yang, A. Ganesh, S. S. Sastry, and Y. Ma, "Robust face recognition via sparse representation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 31, no. 2, pp. 210–227, Feb. 2009.

[11] X. Liu, B. Cheng, S. Yan, J. Tang, T.-S. Chua, and H. Jin, "Label to region by bi-layer sparsity priors," in *Proc. 17th ACM Int. Conf. Multimedia*, 2009, pp. 115–124.

[12] X. Liu, S. Yan, J. Luo, J. Tang, Z. Huang, and H. Jin, "Nonparametric label-to-region by search," in *Proc. Comput. Vis. Pattern Recognit.*, 2010, pp. 3320–3327.

[13] H. Lee, A. Battle, R. Raina, and A. Y. Ng, "Efficient sparse coding algorithms," in *Advances in Neural Information Processing Systems*. Cambridge, MA: MIT Press, 2006, pp. 801–808.

[14] B. Schölkopf, A. J. Smola, and K.-R. Müller, "Kernel principal component analysis," in *Proc. Int. Conf. Artif. Neural Netw.*, 1997, pp. 583–588.

[15] J. Wu and J. M. Rehg, "Beyond the Euclidean distance: Creating effective visual codebooks using the histogram intersection kernel," in *Proc. Int. Conf. Comput. Vis.*, 2009, pp. 630–637.

[16] S. Lazebnik, C. Schmid, and J. Ponce, "Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories," in *Proc. Comput. Vis. Pattern Recognit.*, 2006, pp. 2169–2178.

[17] L. Bo and C. Sminchisescu, "Efficient match kernels between sets of features for visual recognition," in *Advances in Neural Information Processing Systems*. Cambridge, MA: MIT Press, 2009.

[18] S. Gao, I. W. Tsang, and L.-T. Chia, "Kernel sparse representation for image classification and face recognition," in *Proc. Eur. Conf. Comput. Vis.*, 2010, pp. 1–14.

[19] J. Sivic and A. Zisserman, "Video Google: A text retrieval approach to object matching in videos," in *Proc. Int. Conf. Comput. Vis.*, 2003, pp. 1470–1477.

[20] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman, "Object retrieval with large vocabularies and fast spatial matching," in *Proc. Comput. Vis. Pattern Recognit.*, 2007, pp. 1–8.

[21] O. Boiman, E. Shechtman, and M. Irani, "In defense of nearest-neighbor based image classification," in *Proc. Comput. Vis. Pattern Recognit.*, 2008, pp. 1–8.

[22] J. C. van Gemert, J. M. Geusebroek, C. J. Veenman, and A. W. M. Smeulders, "Kernel codebooks for scene categorization," in *Proc. Eur. Conf. Comput. Vis.*, Oct. 2008, pp. 696–709.

[23] J. Wang, J. Yang, K. Yu, F. Lv, and Y. Gong, "Locality-constrained linear coding for image classification," in *Proc. Comput. Vis. Pattern Recognit.*, 2010, pp. 3360–3367.

[24] S. Maji and A. C. Berg, "Max-margin additive classifiers for detection," in *Proc. Int. Conf. Comput. Vis.*, 2009, pp. 1–8.

[25] R. O. Duda, P. E. Hart, and D. G. Stock, *Pattern Classification*, 2nd ed. New York: Wiley, 2001.

[26] J. Ho, M.-H. Yang, J. Lim, K.-C. Lee, and D. J. Kriegman, "Clustering appearances of objects under varying illumination conditions," in *Proc. Comput. Vis. Pattern Recognit.*, 2003, pp. 11–18.

[27] M. Yang, L. Zhang, J. Yang, and D. Zhang, "Robust sparse coding for face recognition," in *Proc. Comput. Vis. Pattern Recognit.*, 2011, pp. 625–632.

[28] Q. Shi, A. Eriksson, A. van den Hengel, and C. Shen, "Is face recognition really a compressive sensing problem?" in *Proc. Comput. Vis. Pattern Recognit.*, 2011, pp. 553–560.

[29] L. Zhang and X. Feng, "Sparse representation or collaborative representation: Which helps face recognition?" in *Proc. Int. Conf. Comput. Vis.*, 2011, pp. 1–8.

[30] V. Guigue, A. Rakotomamonjy, and S. Canu, "Kernel basis pursuit," *Revue d'Intell. Artif.*, vol. 20, no. 6, pp. 757–774, 2006.

[31] V. Roth, "The generalized LASSO," *IEEE Trans. Neural Netw.*, vol. 15, no. 1, pp. 16–28, Jan. 2004.

[32] T. Hesterberg, N. H. Choi, L. Meier, and C. Fraley, "Least angle and ℓ_1 penalized regression: A review," *Stat. Surv.*, vol. 2, pp. 61–93, Nov. 2008.

[33] G. Wang, D. Y. Yeung, and F. H. Lochovsky, "The kernel path in kernelized lasso," in *Proc. AISTATS*, 2007, pp. 1–8.

[34] S. R. Gunn and J. S. Kandola, "Structural modelling with sparse kernels," *Mach. Learn.*, vol. 48, nos. 1–3, pp. 581–591, 2002.

[35] F. Bach and M. Jordan, "Kernel independent component analysis," *J. Mach. Learn. Res.*, vol. 3, pp. 1–48, Jul. 2002.

[36] F. Bach and M. Jordan, "Predictive low-rank decomposition for kernel methods," in *Proc. Int. Conf. Mach. Learn.*, 2005, pp. 1–8.

[37] K. Zhang, I. W. Tsang, and J. T. Kwok, "Improved Nyström low-rank approximation and error analysis," in *Proc. Int. Conf. Mach. Learn.*, 2008, pp. 1232–1239.

[38] P. Drineas and M. W. Mahoney, "On the Nyström method for approximating a Gram matrix for improved kernel-based learning," *J. Mach. Learn. Res.*, vol. 6, pp. 2153–2175, Dec. 2005.

[39] P. Drineas, E. Drinea, and P. Huggins, "An experimental evaluation of a Monte-Carlo algorithm for singular value decomposition," in *Proc. 8th Panhellenic Conf. Inf.*, 2003, pp. 279–296.

[40] B. A. Olshausen and D. J. Fieldt, "Sparse coding with an overcomplete basis set: A strategy employed by V1," *Vis. Res.*, vol. 37, no. 23, pp. 3311–3325, 1997.

[41] S. Maji, A. C. Berg, and J. Malik, "Classification using intersection kernel support vector machines is efficient," in *Proc. Comput. Vis. Pattern Recognit.*, 2008, pp. 1–8.

[42] Y. Huang, K. Huang, Y. Yu, and T. Tan, "Salient coding for image classification," in *Proc. Comput. Vis. Pattern Recognit.*, 2011, pp. 1753–1760.

[43] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *Int. J. Comput. Vis.*, vol. 60, no. 2, pp. 91–110, 2004.

[44] S. Lyu, "Mercer kernels for object recognition with local features," in *Proc. Comput. Vis. Pattern Recognit.*, 2005, pp. 223–229.

[45] D. Haussler, "Convolution kernels on discrete structure," Dept. Comput. Sci., Univ. California at Santa Cruz, Santa Cruz, Tech. Rep. UCS-CRL-99-10, 1999.

[46] Y.-L. Boureau, J. Ponce, and Y. Lecun, "A theoretical analysis of feature pooling in visual recognition," in *Proc. Int. Conf. Mach. Learn.*, 2010, pp. 1–8.

[47] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin, "LIBLINEAR: A library for large linear classification," *J. Mach. Learn. Res.*, vol. 9, pp. 1871–1874, Jun. 2008.

[48] L.-J. Li and L. Fei-Fei, "What, where, and who? Classifying events by scene and object recognition," in *Proc. Int. Conf. Comput. Vis.*, 2007, pp. 1–8.

[49] Z. Lu and H. H. Ip, "Image categorization with spatial mismatch kernels," in *Proc. Comput. Vis. Pattern Recognit.*, 2009, pp. 397–404.

[50] G. Griffin, A. Holub, and P. Perona, "Caltech-256 object category dataset," Dept. Comput. Sci., California Inst. Technology, Pasadena, Tech. Rep. CNS-TR-2007-001, 2007.

[51] A. Georghiades, P. Belhumeur, and D. Kriegman, "From few to many: Illumination cone models for face recognition under variable lighting and pose," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 23, no. 6, pp. 643–660, Jun. 2001.

[52] Z. Lu and H. H. Ip, "Image categorization by learning with context and consistency," in *Proc. Comput. Vis. Pattern Recognit.*, 2009, pp. 2719–2726.

[53] A. Martinez and R. Benavente, "The AR Face database," CVC, Univ. Autònoma de Barcelona, Bellaterra, Barcelona, Tech. Rep. 24, Jun. 1998.

[54] X. He, S. Yan, Y. Hu, P. Niyogi, and H. Zhang, "Face recognition using Laplacianfaces," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 27, no. 3, pp. 328–340, Mar. 2005.

[55] M. Turk and A. Pentland, "Eigenfaces for recognition," in *Proc. Comput. Vis. Pattern Recognit.*, 1991, pp. 1–8.

[56] P. N. Belhumeur, J. P. Hespanha, and D. J. Kriegman, "Eigenfaces vs. fisherfaces: Recognition using class specific linear projection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 19, no. 7, pp. 711–720, Jul. 1997.

[57] C. K. I. Williams and M. Seeger, "Using the Nyström method to speed up kernel machines," in *Advances in Neural Information Processing Systems*. Cambridge, MA: MIT Press, 2000, pp. 682–688.

[58] J. A. K. Suykens and J. Vandewalle, "Least squares support vector machine classifiers," *Neural Process. Lett.*, vol. 9, no. 3, pp. 293–300, 1999.

[59] X. Yuan and S. Yan, "Visual classification with multi-task joint sparse representation," in *Proc. Comput. Vis. Pattern Recognit.*, 2010, pp. 3493–3500.

Shenghua Gao received the B.E. degree from the University of Science and Technology of China, Hefei, China, in 2008. He is currently pursuing the Ph.D. degree with the School of Computer Engineering, Nanyang Technological University, Singapore.

He was a recipient of the Microsoft Research Fellowship in 2010.

Ivor Wai-Hung Tsang received the Ph.D. degree in computer science from the Hong Kong University of Science and Technology, Hong Kong, in 2007.

He is currently an Assistant Professor with the School of Computer Engineering, Nanyang Technological University, Singapore, where he is also the Deputy Director of the Center for Computational Intelligence.

Dr. Tsang was the recipient of the Microsoft Fellowship in 2005, the IEEE TRANSACTIONS ON NEURAL NETWORKS Outstanding 2004 Paper Award in 2006, and the second class prize of the National Natural Science Award 2008, China in 2009. His papers earned him the Best Paper Award at ICTAI'11, the Best Student Paper Award at CVPR'10, and the Best Paper Award from the IEEE Hong Kong Chapter of Signal Processing Postgraduate Forum in 2006.

Liang-Tien Chia received the B.S. and Ph.D. degrees from the Loughborough University of Technology, Loughborough, U.K., in 1990 and 1994, respectively.

He is currently an Associate Professor with the School of Computer Engineering, Nanyang Technological University, Singapore, where he was the Director of the Centre for Multimedia and Network Communications from 2002 to 2007 and the Head of the Division of Computer Communications from 2007 to 2010. He has authored or co-authored more than 100 refereed research papers. His current research interests include internet-related research with emphasis on the semantic web, multimedia understanding through media analysis, annotation, and adaptation, multimodal data fusion, and multimodality ontology for multimedia.