Low-Rank Sparse Coding for Image Classification

Tianzhu Zhang[1,4,5], Bernard Ghanem[1,2], Si Liu[3], Changsheng Xu[4], Narendra Ahuja[1,5]
[1] Advanced Digital Sciences Center of Illinois, Singapore
[2] King Abdullah University of Science and Technology, Saudi Arabia
[3] National University of Singapore, Singapore
[4] Institute of Automation, Chinese Academy of Sciences, P. R. China
[5] University of Illinois at Urbana-Champaign, Urbana, IL USA

Abstract

In this paper, we propose a low-rank sparse coding (LRSC) method that exploits local structure information among features in an image for the purpose of image-level classification. LRSC represents densely sampled SIFT descriptors, in a spatial neighborhood, collectively as low-rank, sparse linear combinations of codewords. As such, it casts the feature coding problem as a low-rank matrix learning problem, which is different from previous methods that encode features independently. This LRSC has a number of attractive properties. (1) It encourages sparsity in feature codes, locality in codebook construction, and low-rankness for spatial consistency. (2) LRSC encodes local features jointly by considering their low-rank structure information, and is computationally attractive. We evaluate the LRSC by comparing its performance on a set of challenging benchmarks with that of 7 popular coding and other state-of-the-art methods. Our experiments show that by representing local features jointly, LRSC not only outperforms the state-of-the-art in classification accuracy but also improves the time complexity of methods that use a similar sparse linear representation model for feature coding [36].

1. Introduction

The bag-of-words (BoW) model is one of the most popular models for feature design. It has been successfully applied to classical computer vision applications, including scene classification [22], image-level object recognition [9, 13], and action recognition [23]. The conventional BoW pipeline for classification consists of five stages: feature extraction and description, codebook design, feature coding, feature pooling, and classifier construction. Recently, different approaches have been proposed to improve the generative property of BoW, that helps it accurately represent images as well as its discriminative power for classification. Despite remarkable progress in this field, there exists significant room for improvement, especially in how local features are encoded in an image.

Given an image, features, such as SIFT [27], HOG [7] and SURF [2], can be densely extracted and encoded with a codebook constructed using K-means clustering. Recently, many different feature coding methods have been proposed including hard-assignment coding (HC) [22], soft-assignment coding (SC*) [33], localized soft-assignment coding (LSC) [25], sparse coding (SCSPM) [36], locality-constrainted linear coding (LLC) [18], Laplacian sparse coding (LScSPM) [11], salient coding (SC) [15], and locality-constrained and spatially regularized coding (LCSRC) [31]. After computing codes for local features, they need to be pooled together to form equal sized feature vectors each representing one image in a dataset. Popular pooling methods include average pooling (e.g. histogram) and max-pooling [36]. To include the spatial layout of local features in an image, Spatial Pyramid Matching (SPM) [22] is usually performed to obtain an image-level representation that can be used to discriminate different categories of objects, scenes, or actions. Using this BoW representation, images can be classified using a plethora of discriminative models such as SVM or Boosting.

Recent work shows that given a visual codebook, the method of encoding local features has significant impact on classification performance. The earliest method is hard-assignment coding (vector quantization) [22], a voting scheme that is simple yet highly sensitive to the selection of codebook. A more robust voting approach is soft-assignment coding [33], which assigns a code coefficient for a particular local feature to each visual word according to their pairwise distance. To improve hard and soft-assignment coding, sparsity is enforced on local feature codes via sparse learning techniques [36]. However, sparse coding is time consuming and usually leads to non-consistent codes [18, 11], i.e. local features with similar descriptors tend to have different sparse codes. To alleviate inconsistency, authors in [37] introduce another coding property, called locality, which encourages that visual words used to represent a local feature be similar to the feature's descriptor itself. This is usually ensured by constructing a feature's codebook from its nearest neighbors in the universal codebook. In fact, several implementa-

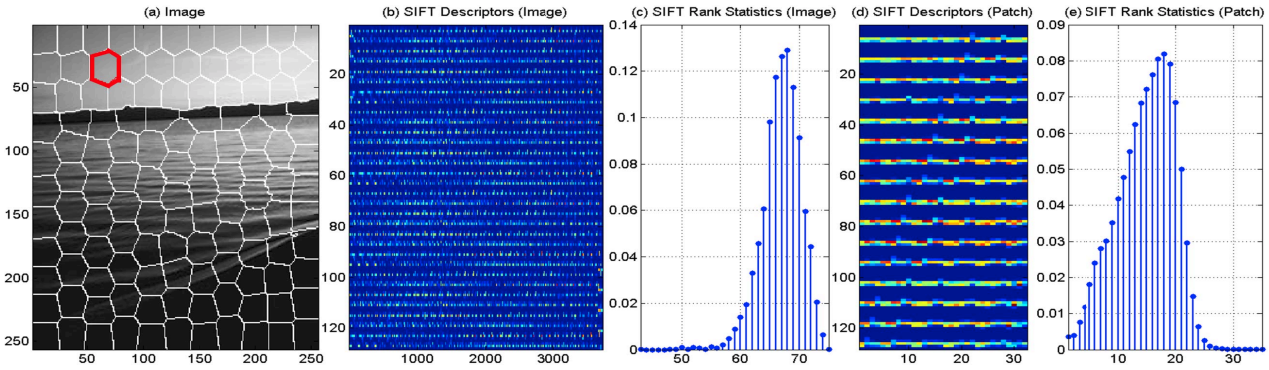

Figure 1. Observing the sparse low-rank property of feature descriptors in natural images. (a) Image segmented into superpixels; (b) All SIFT descriptors densely sampled in (a) concatentated in matrix form; (d) SIFT descriptors from the local region depicted in red color in (a); From Figure (b), we see that SIFT descriptors in the image tend to be sparse and low-rank. This observation holds over thousands of natural images, where the histogram of the rank of their SIFT descriptors is shown in (c). Clearly, the average rank (68) in an image is much smaller than its maximum (128), where each image usually contains thousands of SIFT features. Moreover, this observation transfers locally to superpixels within the image. The histogram of the rank of SIFT descriptors in each superpixel in (a) is plotted in (e). Again, the average rank (18) is smaller than the average number (30) of SIFT features in each superpixel.

tions of locality have been proposed in [18, 25, 15], where each descriptor is coded on locally selected bases. The work in [15] rebrands the locality property as codebook 'saliency'. However, all the aforementioned coding schemes encode local features independently. In Laplacian sparse coding [11], a global similarity between local features is considered to constrain sparsity; however, this method is not only computationally infeasible for large sets of features, but it also disregards the relationship between sparse codes and the spatial layout (or context) of their corresponding features. The property of spatial consistency encourages local features, which are spatially close in an image, to have similar sparse codes and similar supports. The latter implication encourages code consistency among features and suggests that the same visual words represent each local feature in a spatial neighborhood. Very little work has exploited this property in feature coding. In fact, only recently, the spatial layout of local features has been used to select 'optimal' visual words for each feature in an image [31]. This optimal selection is formulated as a labeling problem with a pairwise multi-label MRF energy to be minimized. Despite its awareness of spatial layout, the spatial consistency property is only invoked in the codebook selection process, which is done independently from the coding itself. Although features in a spatial neighborhood are encouraged to have the same set of visual words representing them, their sparse codes are not directly encouraged to be similar or have similar supports w.r.t their 'optimal' bases.

As stated before, maintaining spatial consistency among feature codes enables a more faithful representation of an image and has been shown to improve classification performance. However, many conceivable ways of enforcing such consistency exist. They tend to stem from empirical observations made about spatial relationships between local features in natural images. In fact, we observe that descriptors of detected SIFT points in the same image are not independent and do exhibit a dependency relationship. Figure 1 shows an example of this observation. In Figure 1(b), all SIFT descriptors ($\in \mathbb{R}^{128}$) in the image are concatenated in matrix form. This matrix tends to be sparse and low-rank (refer to Figure 1(d)). In Figure 1(c), we plot the histogram of the rank of this SIFT matrix over thousands of natural images, each containing thousands of dense SIFT features. As can be expected, the average rank of this matrix (68) is much smaller than its maximum possible rank (128). This low-rank sparsity observation is more obvious locally within the same image. By dividing the image into superpixels as shown in Figure 1(a), we observe that the matrix of descriptors for SIFT points in a particular superpixel (denoted in *red*) is also sparse and low-rank as shown in Figure 1(d). In Figure 1(e), we plot the histogram of the rank of SIFT matrices over all superpixels in the image. Clearly, the average rank (18) is much smaller than the average number of SIFT features (30) in each superpixel. Similar observations are also made in [20].

Inspired by the observation and prior work on feature coding, we propose a low-rank sparse coding (LRSC) method that encourages both sparsity and spatial consistency in the coding step of the BoW model. Here, the joint coding of features in a local region is viewed as a low-rank sparse learning problem. Unlike previous methods, we exploit similarities among local features lying in the same spatial neighborhood and, therefore, seek an accurate *joint* representation of these local features w.r.t. a codebook that satisfies the locality property. In LRSC, the codes of local features are sparse and low-rank, which encourages that only a few (but the same) visual words are used to represent all features in a local region. As opposed to sparse coding based image classification methods [36, 18] that handle local features independently, our use of sparse low-rank learning realizes the benefits of a sparse feature representation, while respecting the underlying spatial relationship among local features. Feature codes are computed by solving a sparse low-rank optimization problem, which comprises a sequence of closed form update steps made possible by the Inexact Augmented Lagrange Multiplier (IALM) that guarantees fast convergence.

Contributions: The contributions of this work are threefold. **(1)** We propose a low-rank sparse learning method for feature coding, which is a robust sparse coding method that mines correlations among different local features to obtain better coding results than learning each feature individually. To the best of our knowledge, this is the first work to exploit low-rank sparse learning in feature coding. **(2)** We show that popular sparse coding methods [36, 18] are a special case of our LRSC formulation. **(3)** We learn local feature codes jointly with an efficient IALM method. As a result, LRSC outperforms state-of-the-art coding methods in general, while remaining computationally attractive.

2. Related Work

In this section, we survey commonly used coding schemes. Let $\vec{\mathbf{b}}_i \in \mathbb{R}^d$ denote a visual word in the codebook, where d is the dimensionality of a local feature. The total number of visual words is n. Matrix $\mathbf{B} = \left[\vec{\mathbf{b}}_1, \vec{\mathbf{b}}_2, \cdots, \vec{\mathbf{b}}_n\right]$ denotes a visual codebook or a set of basis vectors. Let $\vec{\mathbf{x}}_i \in \mathbb{R}^d$ be the i^{th} local feature in an image. Let $\vec{\mathbf{z}}_i \in \mathbb{R}^n$ be the code of $\vec{\mathbf{x}}_i$, with $\mathbf{z_{ij}}$ being the coefficient w.r.t. word $\vec{\mathbf{b}}_j$.

Hard-assignment coding (HC) [22]: For a local feature $\vec{\mathbf{x}}_i$, there is one and only one nonzero coding coefficient. It corresponds to the nearest visual word subject to a predefined distance. When we adopt Euclidean distance,

$$\mathbf{z_{ij}} = \begin{cases} 1 & \text{if} \quad j = \underset{j=1,\ldots,n}{\arg\min} \left\| \vec{\mathbf{x}}_i - \vec{\mathbf{b}}_j \right\|_2^2 \\ 0 & \text{otherwise} \end{cases}$$

Soft-assignment coding (SC*) [33]: The j-th coding coefficient represents the degree of membership of a local feature $\vec{\mathbf{x}}_i$ to the j^{th} visual word, where α is the smoothing factor controlling the softness of the assignment. Note that all n visual words are used in computing $\mathbf{z_{ij}}$.

$$\mathbf{z_{ij}} = \frac{\exp(-\alpha \|\vec{\mathbf{x}}_i - \vec{\mathbf{b}}_j\|_2^2)}{\sum_{k=1}^{n} \exp(-\alpha \|\vec{\mathbf{x}}_i - \vec{\mathbf{b}}_k\|_2^2)}$$

Localized soft-assignment coding (LSC) [25]: The basic idea is to adopt the k visual words in the neighborhood of a local feature to refine the soft-assignment coding [33].

$$\mathbf{z_{ij}} = \frac{\exp(-\alpha \|\vec{\mathbf{x}}_i - \vec{\mathbf{b}}_j\|_2^2)}{\sum_{k=1}^{n} \exp(-\alpha \|\vec{\mathbf{x}}_i - \vec{\mathbf{b}}_k\|_2^2)}, \vec{\mathbf{b}}_k \in \mathbf{N}_k(\vec{\mathbf{x}}_i)$$

Sparse coding (SCSPM) [36]: It represents a local feature $\vec{\mathbf{x}}_i$ by a linear combination of a sparse set of basis vectors in the codebook. The coefficient vector $\vec{\mathbf{z}}_i$ is obtained by solving an ℓ_1-norm regularized problem,

$$\vec{\mathbf{z}}_i = \arg\min \|\vec{\mathbf{x}}_i - \mathbf{B}\vec{\mathbf{z}}_i\|_2^2 + \lambda \|\vec{\mathbf{z}}_i\|_1$$

Locality-constrained linear coding (LLC) [18]: Unlike sparse coding, LLC enforces codebook locality instead of sparsity. This leads to smaller coefficients for basis vectors farther away from $\vec{\mathbf{x}}_i$. The code $\vec{\mathbf{z}}_i$ is computed by solving the following regularized least-squares program,

$$\vec{\mathbf{z}}_i = \underset{\mathbf{1}^T \vec{\mathbf{z}}_i = 1}{\arg\min} \|\vec{\mathbf{x}}_i - \mathbf{B}\vec{\mathbf{z}}_i\|_2^2 + \lambda \|\mathbf{d_i} \odot \vec{\mathbf{z}}_i\|_2^2$$

where $\mathbf{d_i} = \exp(dist(\vec{\mathbf{x}}_i, \mathbf{B})/\delta)$, $dist(\vec{\mathbf{x}}_i, \mathbf{B}) = (dist(\vec{\mathbf{x}}_i, \vec{\mathbf{b}}_1), dist(\vec{\mathbf{x}}_i, \vec{\mathbf{b}}_2), \ldots, dist(\vec{\mathbf{x}}_i, \vec{\mathbf{b}}_n))^T$, and $dist(\vec{\mathbf{x}}_i, \vec{\mathbf{b}}_j)$ denotes the ℓ_2 distance between $\vec{\mathbf{x}}_i$ and each $\vec{\mathbf{b}}_j$. δ is used for adjusting the weight decay speed for the locality adaptor. In [18], an approximation is proposed to improve its computational efficiency.

Laplacian sparse coding (LScSPM) [11]: It is the first method that improves the consistency of sparse coding, by encouraging similar local features in the dataset to have similar sparse codes. This is done by adding a graph regularization term to the LASSO. Codebook learning and sparse coding are done iteratively.

$$(\mathbf{B}, \mathbf{Z}) = \underset{\mathbf{B},\mathbf{Z}}{\arg\min} \|\mathbf{X} - \mathbf{BZ}\|_2^2 + \lambda \textstyle\sum_i \|\vec{\mathbf{z}}_i\|_1 + \beta tr(\mathbf{ZLZ^T})$$
$$s.t. \quad \|\vec{\mathbf{b}}_j\|^2 \leq 1, \ \ \forall\, j = 1, \ldots, n$$

Here, $\mathbf{L}$ is the Laplacian of the graph that encodes the relationship between local features. Due to the extremely large number of features in a dataset, constructing the Laplacian matrix and learning sparse codes simultaneously is computationally infeasible. Some heuristic measures are taken to moderately improve its computational complexity.

Salient coding (SC) [15]: This is an alternative to sparse coding. It exploits codebook locality by setting the code to a "saliency" degree based on the nearest codebook bases $\vec{\mathbf{b}}_j$ to $\vec{\mathbf{x}}_i$. Here, $\varphi(.)$ is a monotonically decreasing function and $\{\hat{\vec{\mathbf{b}}}_m\}_{m=1,\ldots,k}$ is the set of k-nearest bases to $\vec{\mathbf{x}}_i$.

$$\mathbf{z}_{i,j} = 1 - \varphi\left(\frac{\|\vec{\mathbf{x}}_i - \vec{\mathbf{b}}_j\|_2^2}{\frac{1}{k-1}\sum_{m \neq j}^{k} \|\vec{\mathbf{x}}_i - \hat{\vec{\mathbf{b}}}_m\|_2^2}\right)$$

Locality-constrained and spatially regularized coding (LCSRC) [31]: The spatial layout of local features in the same image is used to select "optimal" bases for each local feature. It assumes that local feature $\vec{\mathbf{x}}_p$ should have similar bases as its nearest neighbors $\vec{\mathbf{x}}_q$. This is done by solving a pairwise multi-label MRF problem. Once bases are selected for local features, their codes can be computed by using any of the previous coding methods.

Most of the aforementioned coding schemes (except for LScSPM) produce feature codes independently. Although LScSPM [11] adopts a global similarity between local features, it ignores local spatial contextual information [16, 39, 40] and is computationally expensive. LCSRC [31] makes use of the spatial layout of local features in the same image. However, it only does so to constrain codebook selection. It fails to directly enforce consistency on codes themselves.

To the best of our knowledge, the proposed low-rank sparse coding (LRSC) method is the first to introduce spatial consistency and joint feature coding explicitly in the coding step of the BoW model. In the next section, we provide a detailed description of LRSC coding.

3. Low-Rank Sparse Coding (LRSC)

Here, we give a detailed description of our local feature coding method that makes use of low-rank sparse learning.

3.1. Low-Rank Sparse Representation

As seen in Figure 1, SIFT descriptors tend to be collectively sparse and low-rank across natural images and specifically in spatial neighborhoods of the same image. However, many existing methods [22, 36, 18, 25, 15] ignore this information and encode features independently. In this paper, we formulate local feature coding as a low-rank sparse learning problem, which encourages sparsity and low-rankness locally among features in the image. Since the low-rank sparsity property is more evident locally, we apply low-rank sparse learning to code features in the same region of an image, by dividing an image into homogeneous superpixels. Without loss of generality, we use the SLIC segmentation algorithm [1]. We divide each image into around 150 coherent superpixels. The details and effects of segmentation will be discussed in Section 5.1. Note that the solution is general and not tied to any specific image segmentation algorithm.

Following many coding methods [22, 36, 18, 25, 15], LRSC densely samples SIFT features in an image. Each region contains n local features, whose observations (SIFT descriptors) are concatentated in matrix form as: $\mathbf{X} = [\vec{\mathbf{x}}_1, \vec{\mathbf{x}}_2, \cdots, \vec{\mathbf{x}}_n]$. Each column is a local feature point in $\mathbb{R}^d$, where $d = 128$ usually. Given a codebook, $\mathbf{D} = \left[\vec{\mathbf{d}}_1, \vec{\mathbf{d}}_2, \cdots, \vec{\mathbf{d}}_m\right]$, in the noiseless case, each local feature $\vec{\mathbf{x}}_i$ is represented as a linear combination $\vec{\mathbf{z}}_i$ of elements forming the codebook $\mathbf{D}$, such that $\mathbf{X} = \mathbf{DZ}$.

We base the formulation of LRSC on the following observations. **(a)** Because features are densely sampled in a local region, they tend to have similar descriptors, as exemplified in Figure 1. Consequently, their representations w.r.t. to $\mathbf{D}$ should also be similar. Therefore, the resulting representation matrix $\mathbf{Z}$ is expected to be low-rank. More formally, we see that since $\mathbf{D}$ is an overcomplete full rank matrix, then rank($\mathbf{DZ}$) is equal to rank($\mathbf{Z}$). Therefore, if rank($\mathbf{X}$) is low (as shown in Figure 1), then rank($\mathbf{Z}$) should also be low too (as shown in Figure 2). **(b)** For an overcomplete dictionary $\mathbf{D}$, linear feature representations w.r.t. $\mathbf{D}$ tend to be sparse. In other words, only a few elements of $\mathbf{D}$ are required to reliably represent a local feature $\vec{\mathbf{x}}_i$ or equivalently only a few nonzero coefficients exist in its representation $\vec{\mathbf{z}}_i$. In fact, sparse feature coding has been shown to be quite helpful in image classification [36, 26, 18]. We combine **(a)** and **(b)** to formulate the problem mathematically in Eq 1, whose solution is described in Section 4. The nuclear norm $\|\mathbf{Z}\|_*$ and the sparsity inducing ℓ_1 norm $\|\mathbf{Z}\|_{1,1} = \sum_{i=1}^{n} \|\vec{\mathbf{z}}_i\|_1$ are convex approximations to the rank function and ℓ_0 norm, respectively. λ_1 and λ_2 quantify the tradeoff between sparsity and low-rankness in the feature codes.

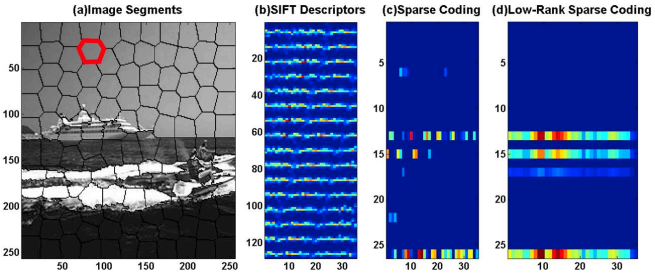

Figure 2. A feature coding example in a local region. (a) Image partition results; (b) All SIFT descriptors in the local region depicted in red in (a); (c) and (d) are coding results produced by SCSPM [36] and LRSC. From (c), we see that the local features have inconsistent codes, i.e. their features are similar but their codes and the supports of their codes are not. This is because SCSPM solves the coding problem for each feature independently. However, the codes learnt by LRSC are jointly sparse, i.e. a few (but the same) visual words are used to represent all the local features together, which renders the codes consistent and more robust to noise. The dictionary is obtained by using the locality property.

$$\min_{\mathbf{Z}} \frac{1}{2} \|\mathbf{X} - \mathbf{DZ}\|_F^2 + \lambda_1 \|\mathbf{Z}\|_* + \lambda_2 \|\mathbf{Z}\|_{1,1} \qquad (1)$$

3.2. Discussion

As stated earlier, many feature coding schemes exist in the literature. In HC [22] and SC* [33], different voting schemes are adopted to obtain $\vec{\mathbf{z}}_i$ for each local feature. SCSPM [36] improves upon these methods by enforcing sparsity in $\vec{\mathbf{z}}_i$. However, solving an ℓ_1 problem for each local feature independently is computationally expensive, especially for large codebooks. In LLC [18], LSC [25], and SC [15], locality in codebook selection is adopted and better performance is obtained. In LScSPM [11], a global similarity among features is adopted to consider the relationship among feature points in feature space. However, it incurs a significantly high computation cost and ignores spatial relationships between features in the same image. Recently, spatial consistency has been successfully adopted by LCSRC [31] for feature coding. *It ignores the fact that features in a local region not only have similar codebooks, but also similar representations.* Clearly, features with similar bases coded independently may have different representations. The LRSC method we propose here is aimed at simultaneously achieving all three properties for image classification: sparsity, locality, and spatial consistency. In Figure 2, we show an example of how LRSC compares with traditional sparse coding (SCSPM) [36]. Clearly, the columns of $\mathbf{Z}$ generated by LRSC are jointly sparse, i.e. a few (but the same) visual words are used to represent all the local features. This exemplifies how both the sparsity and low-rank properties are satisfied under LRSC. This is not the case for SCSPM, which is known to produce inconsistent codes, especially for features present in

the same spatial neighborhood. The following three observations explain how LRSC is related to other coding schemes.

- Sparsity: When $\lambda_2 \neq 0$, LRSC leads to sparse codes. When $\lambda_1 = 0$ (i.e. spatial consistency is not considered), our method degenerates into SCSPM.
- Locality: When encoding features in a local region, we adopt locality in selecting the codebook $\mathbf{D}$. Similar to LLC and LSC, we construct $\mathbf{D}$ from elements in the universal codebook that are nearest to each local feature.
- Spatial Consistency: By enforcing the low-rank property in a local region, spatial information is encoded and features are constrained to have similar codes. In comparison, LCSRC [31] incorporates spatial consistency in selecting an 'optimal' codebook for each feature separately and then computes feature codes independently in the image. Using this method, there is no direct guarantee that features in a local region have similar codes.

4. Optimization

In Eq (1), the cost function has two convex and non-smooth regularizers (sparse $\|.\|_1$ regularizer or low-rank $\|.\|_*$ regularizer), which makes solving it efficiently non-trivial. In order to handle these two regularizers independently, we introduce two slack variables and add two equality constraints as in Eq (2).

$$\min_{\mathbf{Z}_{1-3}} \ \frac{1}{2}\|\mathbf{X}-\mathbf{D}\mathbf{Z}_3\|_F^2 + \lambda_1\|\mathbf{Z}_1\|_* + \lambda_2\|\mathbf{Z}_2\|_{1,1} \quad (2)$$
$$\text{such that: } \mathbf{Z_3} = \mathbf{Z_1};\ \ \mathbf{Z_3} = \mathbf{Z_2}$$

This transformed problem can be minimized using the conventional Inexact Augmented Lagrange Multiplier (IALM) method that has attractive quadratic convergence properties and is extensively used in matrix rank minimization problems [29]. IALM is an iterative method that augments the Lagrangian function with quadratic penalty terms. This allows closed form updates for each of the variables. The updates are closed form due to the identities in Eq (3,4), where $\mathcal{S}_\lambda(\mathbf{A}_{ij}) = \text{sign}(\mathbf{A}_{ij})\max(0, |\mathbf{A}_{ij}| - \lambda)$ is the soft-thresholding operator and $\mathcal{J}_\lambda(\mathbf{A}) = \mathbf{U}_A\mathcal{S}_\lambda(\mathbf{\Sigma}_A)\mathbf{V}_A^T$ is the singular value soft-thresholding operator with $\mathbf{A} = \mathbf{U}_A\mathbf{\Sigma}_A\mathbf{V}_A^T$ being the SVD of $\mathbf{A}$.

$$\mathbf{X}^* = \arg\min \ \|\mathbf{X}-\mathbf{A}\|_F^2 + 2\lambda\|\mathbf{X}\|_{1,1} = \mathcal{S}_\lambda(\mathbf{A}) \quad (3)$$
$$\mathbf{X}^* = \arg\min \ \|\mathbf{X}-\mathbf{A}\|_F^2 + 2\lambda\|\mathbf{X}\|_* = \mathcal{J}_\lambda(\mathbf{A}) \quad (4)$$

By introducing augmented lagrange multipliers to incorporate the equality constraints into the cost function, we obtain the Lagrangian function in Eq (5) that we show, in what follows, can be minimized through a sequence of simple closed form update operations.

$$\begin{aligned} L(\mathbf{Z}_{1-3}) = {} & \frac{1}{2}\|\mathbf{X}-\mathbf{D}\mathbf{Z}_3\|_F^2 + \lambda_1\|\mathbf{Z}_1\|_* + \lambda_2\|\mathbf{Z}_2\|_{1,1} \\ & + tr\left[\mathbf{Y}_1^T(\mathbf{Z}_3-\mathbf{Z}_1)\right] + \frac{u_1}{2}\|\mathbf{Z}_3-\mathbf{Z}_1\|_F^2 \\ & + tr\left[\mathbf{Y}_2^T(\mathbf{Z}_3-\mathbf{Z}_2)\right] + \frac{u_2}{2}\|\mathbf{Z}_3-\mathbf{Z}_2\|_F^2 \end{aligned} \quad (5)$$

$\mathbf{Y}_1$ and $\mathbf{Y}_2$ are lagrange multipliers, and $u_1 > 0$ and $u_2 > 0$ are two penalty parameters. The above problem can by minimized by either exact or inexact ALM algorithms [24]. For efficiency, we choose the inexact ALM. Its convergence properties can be proven similar to those in [24]. In fact, IALM is an iterative algorithm that solves for each variable in a blockwise coordinate descent fashion. In other words, each iteration of IALM involves updating each variable one-at-a-time, with the other variables fixed to their most recent values. Consequently, we obtain four update steps corresponding to the four sets of variables we need to optimize for. Note that all steps have closed form solutions.

Step 1: [Update $\mathbf{Z}_1$] This requires solving the following problem as shown in Eq (6).

$$\begin{aligned} & \mathbf{Z}_1^* = \arg\min_{\mathbf{Z}_1} \frac{\lambda_1}{u_1}\|\mathbf{Z}_1\|_* + \frac{1}{2}\|\mathbf{Z}_1 - (\mathbf{Z}_3 + \frac{1}{u_1}\mathbf{Y}_1)\|_F^2 \\ \Rightarrow & \boxed{\mathbf{Z}_1^* = \mathcal{J}_{\frac{\lambda_1}{u_1}}(\mathbf{Z}_3 + \tfrac{1}{u_1}\mathbf{Y}_1)} \end{aligned} \quad (6)$$

Step 2: [Update $\mathbf{Z}_2$] This is done by solving Eq (7).

$$\begin{aligned} & \mathbf{Z}_2^* = \arg\min_{\mathbf{Z}_2} \frac{\lambda_2}{u_2}\|\mathbf{Z}_2\|_{1,1} + \frac{1}{2}\|\mathbf{Z}_2 - (\mathbf{Z}_3 + \frac{1}{u_2}\mathbf{Y}_2)\|_F^2 \\ \Rightarrow & \boxed{\mathbf{Z}_2^* = \mathcal{S}_{\frac{\lambda_2}{u_2}}(\mathbf{Z}_3 + \tfrac{1}{u_2}\mathbf{Y}_2)} \end{aligned} \quad (7)$$

Step 3: [Update $\mathbf{Z}_3$] This is done by solving Eq (8), whose solution is shown in Eq (9).

$$\begin{aligned} \mathbf{Z}_3^* = \arg\min_{\mathbf{Z}_3} \ & \frac{1}{2}\|\mathbf{X}-\mathbf{D}\mathbf{Z}_3\|_F^2 + tr[\mathbf{Y}_1^t(\mathbf{Z}_3-\mathbf{Z}_1)] \\ & + \frac{u_1}{2}\|\mathbf{Z}_3-\mathbf{Z}_1\|_F^2 + tr[\mathbf{Y}_2^t(\mathbf{Z}_3-\mathbf{Z}_2)] + \frac{u_2}{2}\|\mathbf{Z}_3-\mathbf{Z}_2\|_F^2 \end{aligned} \quad (8)$$

$$\Rightarrow \boxed{\mathbf{Z}_3^* = \left(\mathbf{D}^T\mathbf{D} + u_1\mathbf{I} + u_2\mathbf{I}\right)^{-1}\mathbf{G}}, \quad (9)$$

where $\mathbf{G} = \mathbf{D}^T\mathbf{X} - \mathbf{Y}_1 - \mathbf{Y}_2 + u_1\mathbf{Z}_1 + u_2\mathbf{Z}_2$.

Step 4: [Update Multipliers] They are updated in Eq (10), where $\rho > 1$ is a user-defined constant.

$$\begin{cases} \mathbf{Y}_1 = \mathbf{Y}_1 + u_1(\mathbf{Z}_3-\mathbf{Z}_1); \mathbf{Y}_2 = \mathbf{Y}_2 + u_2(\mathbf{Z}_3-\mathbf{Z}_2) \\ u_1 = \rho u_1;\ u_2 = \rho u_2 \end{cases} \quad (10)$$

Computational Complexity: The convergence of IALM algorithm is reached when the change in objective function or solution $\mathbf{Z}$ is below a user-defined threshold $\epsilon = 10^{-3}$.

Empirically, we find that our IALM algorithm is insensitive to a large range of ϵ values. In our implementation, $u_1 = u_2$. The computational bottleneck of LRSC lies in the SVD of matrix $\mathbf{Z}$ in Step 1. Since $\mathbf{Z}$ is low-rank and rectangular, its SVD can be computed efficiently with time complexity $\mathcal{O}(mnr)$, where r is its rank such that $r \leq \sqrt{\min(m,n)}$. Because r is usually small compared to m and n and the matrix inversion in Step 3 can be done by the eigenvalue decomposition of $\mathbf{D}^T\mathbf{D}$ only one time at the start of the optimization, the overall computational complexity of LRSC is $\mathcal{O}\left(m^3 + mn\epsilon^{-0.5}\right)$, where the number of IALM iterations is $\mathcal{O}\left(\epsilon^{-0.5}\right)$. In comparison, SCSPM solves n ℓ_1 minimization (LASSO) problems independently and thus has a time complexity of $O(m^2nd)$, which is signficantly slower than our coding method. In practise, we observe that LRSC is usually about 4 times faster than SCSPM.

5. Experimental Results

In this section, we experimentally assess the generality of our LRSC method by evaluating its performance on two different tasks: scene classification and objection recognition. The effectiveness and efficiency of LRSC are validated by a comparison with 7 popular coding methods and other state-of-the-art approaches where applicable.

Datasets: For the two tasks, LRSC is evaluated on four well known benchmarks, intensively used in the literature: Scene-13 [10], Caltech-101 [9], Caltech-256 [13], and UIUC 8-Sport [23].

Baseline Methods: We compare LRSC to two types of image classification methods in the literature: (1) methods relevant to feature coding that use the same BoW pipeline for image classification but only differ in how coding is performed and (2) other well-known classification methods that do not necessarily conform to the BoW pipeline. Direct evaluation of LRSC is made by comparing it to methods of type (1), since all stages of the BoW pipeline (e.g. feature extraction and classification) are kept the same and only the coding stage is varied. For completeness, we compare the performance of LRSC against that of type (2) methods, even though the feature, representation, and classification schemes used there are quite different. We include 7 recent and state-of-the-art type (1) methods, which are denoted as: HC [22], LSC [25], SCSPM [36], LLC [18], LScSPM [11], SC [15], and LCSRC [31]. On UIUC 8-Sport, Scene-15, and Caltech-101, the baseline results for these methods are borrowed from [31]. On the other data sets, we implement these methods using publicly available source codes or binaries provided by the authors and run them with default parameters. For each dataset, we also include state-of-the-art type (2) methods and report their results.

Implementation Details: For fair comparison with type (1) methods, we fix all stages of the BoW classification pipeline except for the feature coding stage. As reference, we follow the experimental setup in [31] for all our experiments.

Table 1. Classification accuracies on the UIUC 8-Sport data set.

Methods	Accuracies (%)	Methods	Accuracies (%)
HC [22]	79.98 ± 1.67	LScSPM [11]	85.31 ± 0.51
SCSPM [36]	82.74 ± 1.46	SC [15]	85.44 ± 1.54
LLC [18]	81.77 ± 1.51	LCSRC [31]	87.23 ± 1.14
LSC [25]	82.79 ± 2.01	LRSC	$\mathbf{88.17 \pm 0.85}$

For completeness, we briefly describe this setup next. *(1) Image resize:* Similar to previous methods [36, 25, 18, 11, 31], images are downsized to no more than 300×300 pixels for Scene-13, Caltech-101, and Caltech-256, and 400×400 pixels for UIUC 8-Sport, respectively. *(2) Dense local features:* SIFT descriptors [27] with dimension $d = 128$ are extracted from 16×16 pixel patches densely sampled from each image on a grid with a 4 pixel stepsize. *(3) Codebook:* The universal codebook is obtained using K-means on a randomly selected subset of SIFT descriptors ($200K$) in the training set. As in [31], the codebook size depends on the size of the dataset: 1024 for Scene-13, Caltech-101, and UIUC 8-Sport and 4096 for Caltech-256. As discussed in [36, 18, 6], increasing the codebook size can improve the performance. Due to the locality property of the dictionary discussed in Section 3.2, our algorithm will incur a slightly higher computational cost to find the nearest neighbors in codebook for each feature point. Therefore, our algorithm can retain a good performance level even for large codebook sizes. For a fair comparison, we adopt the setup of [31]. *(4) Local region:* SLIC segmentation [1] is adopted to segment images into multiple superpixels. SLIC has three parameters: MinRegionSize, regionSize, and regularizer, which are set to be 100, 24, and 1, respectively. The details are discussed in Section 5.1. *(5) Pooling:* Max-pooling is performed. To include spatial layout information, SPM [22] with 3 levels: 1×1 , 2×2 and 4×4 is adopted. The weight for each layel is the same. *(6) Classifier:* a one-vs-all linear SVM classifier is used, since it has been shown to achieve state-of-the-art classification performance when paired with max-pooling [36, 25, 18, 31].

5.1. UIUC 8-Sport Data Set

UIUC 8-Sport [23] contains 1792 images and 8 categories for image-based event classification. These 8 categories are badminton, bocce, croquet, polo, rock climbing, rowing, sailing and snow boarding, and the size of each category ranges from 137 to 250. Following the standard setting for this data set, we use 10 random splits of the data, we randomly select 70 training images and 60 test images for each category. The classification accuracy is reported in Table 1, which shows the average (and standard deviation) results of the state-of-the-art coding approaches and the proposed LRSC method. As we can see, SCSPM is much better than the classic HC method, which shows sparsity is helpful for image classification. In LLC, LSC, and SC, adding locality can also improve the classification accuracy. Results of LScSPM show that the relationships among features in their $d-$dimensional feature space improves classification further.

Table 2. The influence of image partition to LRSC.

Partition	20×20	30×30	40×40
Accuracies (%)	**86.2 ± 0.81**	**88.17 ± 0.85**	**87.3 ± 0.91**

In LCSRC and our LRSC, adding local spatial information improves classification accuracy significantly as compared to the first six methods, and our LRSC has a moderate improvement over the state-of-the-art feature coding methods. Compared with other state-of-the-art non-coding methods as shown in Table 1, our LRSC is much better than GIST [28] (63.88), [19] (83.54±1.13), [23] (73.4), [30] (86.25), and has about 2% improvement compared with the latest work [30].

Since our LRSC is used to encode features in local regions (superpixels), we now study the influence of image partition on LRSC by comparing its performance under different partition settings. In this work, we adopt the SLIC approach [1] to segment images into multiple homogeneous patches (superpixels). As for SLIC parameters, *MinRegionSize* and *regularizer* are set to 100 and 1, respectively. For comparison, we vary *regionSize* (nominal size of the superpixels) between three values: 20, 30, and 40. The corresponding results are reported in Table 2. There is only a slight difference among the three settings. Following the best result, we set *regionSize* to 30. Clearly, optimizing the selection of partition parameters for each task and dataset will improve LRSC performance. However, we leave this for future work.

In Table 3, the runtime for all coding methods on the same image is reported. For one 300×400 image with 108 segments, 6984 SIFT descriptors are extracted. When all features are coded with a 1024 codebook, LRSC is computationally much faster than SCSPM [36] because our LRSC encodes local features jointly, which is much more efficient than SCSPM encoding features independently (6984 ℓ_1 minimization problems). LRSC is also comparable with HC [22], SC [15], LLC [18], and LSC [25], which do not perform expensive optimization operations. We could not compare against the runtime of LScSPM and LCSRC because their source codes were not available. But, we expect LRSC to be faster, since LCSRC need solve an expensive multi-label MRF problem. All experiments are done using MATLAB on a 2.66GHZ Intel Core2 Duo PC with 18GB RAM.

5.2. Scene-13 Data Set

Scene-13 [10] consists of 3859 images each belonging to one of 13 categories, which contain 200 to 400 images each. The categories vary from outdoor scenes like mountain and forest to indoor environments like living room and kitchen. Following the standard setup, we use 10 random splits of the data, while considering 100 random images per class for training and the rest for testing. The comparative results are shown in Table 4. LRSC performs best among all the feature coding methods and has about 2% improvement. Compared with other state-of-the-art methods [19, 10, 4], LRSC performs much better ([19] (83.54 ± 1.13), [10] (65.2) and has about 2% improvement.

Table 3. Runtime of different coding methods on a 300×400 image with 6984 SIFT descriptors. The codebook is 1024, and the number of superpixels is 108.

Methods	Time (seconds)	Methods	Time (seconds)
HC [22]	1.66	LScSPM [11]	-
SCSPM [36]	8.27	SC [15]	1.73
LLC [18]	1.65	LCSRC [31]	-
LSC [25]	1.71	LRSC	2.33

Table 4. Classification accuracies on the Scene-13 data set.

Methods	Accuracies (%)	Methods	Accuracies (%)
HC [22]	77.20 ± 0.41	LScSPM [11]	-
SCSPM [36]	83.14 ± 0.45	SC [15]	82.11 ± 0.34
LLC [18]	83.25 ± 0.36	LCSRC [31]	-
LSC [25]	83.33 ± 0.44	LRSC	**85.13 ± 0.53**

5.3. Caltech-101 Data Set

Caltech-101 [9] contains 9144 images in 101 classes including animals, vehicles, flowers, etc, with high shape variability. The number of images per category varies from 31 to 800. Following the standard experimental setting, we use 10 random splits of the data, while considering 30 random images per class for training and the rest for testing. The average classification rates are reported in Table 5. From these results, we see that LRSC performs best among the existing methods. As compared to the sparse coding methods SCSPM and LLC, LRSC's performance is much better, since it makes a 3% improvement. It also registers about 2% improvement over LCSRC. As such, we conclude that exploiting spatial consistency directly in the coding stage improves classification performances by 3% on average. We could not compare against LScSPM because its source code was not available. We also compare our results to the state of art using one type of descriptors on Caltech-101. Our LRSC is better than [3] (70.4), [17] (69.6), [38] (66.2 ± 0.5), [5] (75.1 ± 0.9), and is comparable to [5] (75.7 ± 1.1), which adopts kernel SVM as the classifier. In [6] (77.3 ± 0.6), and [8] (80.3 ± 1.2), they adopt macrofeatures, cross-validation to tune parameters, and kernel SVM, respectively, and show much better performance than our LRSC. Note that better performance has been reported with multiple descriptor types (e.g., methods using multiple kernel learning have achieved 77.7% ± 0.3 [12], 78.0% ± 0.3 [14, 34], and 84.3% [35]), or subcategory learning (83% [32]).

5.4. Caltech-256 Data Set

Caltech-256 [13] contains 256 categories as well as a background class. The number of images is 29780 with much higher intra-class variability and higher object location variability as compared to Caltech-101, in which the objects are often in the center of image. Clearly, Caltech-256 is a very challenging data set for object recognition. Following

Table 5. Classification accuracies on the Caltech-101 data set.

Methods	Accuracies (%)	Methods	Accuracies (%)
HC [22]	69.43 ± 0.52	LScSPM [11]	-
SCSPM [36]	72.20 ± 1.30	SC [15]	69.55 ± 0.83
LLC [18]	71.67 ± 0.86	LCSRC [31]	73.23 ± 0.81
LSC [25]	72.58 ± 1.08	LRSC	**75.02 ± 0.74**

Table 6. Classification accuracies on the Caltech-256 data set.

Methods	Accuracies (%)	Methods	Accuracies (%)
HC [22]	21.82 ± 0.22	LScSPM [11]	35.74 ± 0.10
SCSPM [36]	34.02 ± 0.35	SC [15]	34.60 ± 0.27
LLC [18]	37.41 ± 0.21	LCSRC [31]	-
LSC [25]	38.15 ± 0.26	LRSC	$\mathbf{41.04 \pm 0.23}$

the standard experimental setting, we use 10 random splits of the data, while considering 30 random images per class for training and the rest for testing and list the average classification rates in Table 6. From this table, we see that our LRSC method outperforms the other coding methods on this data set, and makes about 3% improvement. Compared with other state-of-the-art methods, our LRSC is also much better than [21] (36.3), [8] (38.1 ± 0.6) and [3](1 desc) (37.0), and is is also comparable to Boureau et al. [6] (41.7 ± 0.8) with macrofeatures and cross-validation. In addition, NBNN (5 desc) [3] (42.0) and Todorovic et al. [32] (49.5) show much better performance due to the use of multiple features.

6. Conclusion

In this paper, we present a new coding technique for local features that employs low-rank sparse learning. This method exploits sparsity in individual codes, locality in codebook selection, and low-rankness in constraining sparse codes belonging to the same spatial neighborhood. Although the selection of spatial neighborhoods (superpixels) might not be optimal, our extensive results show that our method improves upon the state-of-the-art and increases classification accuracy on several benchmarks. For future work, we will systematically study how image partition can be combined with low-rank sparse coding in one unified framework.

Acknowledgment

This study is 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). Narendra Ahuja was supported, in part, by the Office of Naval Research under grant N00014-12-1-0259.

References

[1] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Susstrunk. Slic superpixels. In *Technical report, EPFL*, 2010.

[2] H. Bay, T. Tuytelaars, and L. V. Gool. Surf: Speeded up robust features. In *ECCV*, 2006.

[3] O. Boiman, I. Rehovot, E. Shechtman, and M. Irani. In defense of nearest-neighbor based image classification. In *CVPR*, 2008.

[4] A. Bosch, A. Zisserman, and X. Muñoz. Scene classification via plsa. In *ECCV*, 2006.

[5] Y.-L. Boureau, F. Bach, Y. LeCun, and J. Ponce. Learning mid-level features for recognition. In *CVPR*, 2010.

[6] Y.-L. Boureau, N. L. Roux, F. Bach, J. Ponce, and Y. LeCun. Ask the locals: multi-way local pooling for image recognition. In *ICCV*, 2011.

[7] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *CVPR*, 2005.

[8] O. Duchenne, A. Joulin, and J. Ponce. A graph-matching kernel for object categorization. In *ICCV*, 2011.

[9] L. Fei-Fei, R. Fergus, and P. Perona. Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. In *CVIU*, 2007.

[10] L. Fei-Fei and P. Perona. A bayesian hierarchical model for learning natural scene categories. In *CVPR*, 2005.

[11] S. Gao, I. Tsang, L. Chia, and P. Zhao. Local features are not lonely - laplacian sparse coding for image classification. In *CVPR*, 2010.

[12] P. Gehler and S. Nowozin. On feature combination formulticlass object classification. In *ICCV*, 2009.

[13] G. Griffin, A. Holub, and P. Perona. Caltech-256 object category dataset. In *techreport*, 2007.

[14] http://www.robots.ox.ac.uk/ vgg/software/MKL/.

[15] Y. Huang, K. Huang, Y. Yu, and T. Tan. Salient coding for image classification. In *CVPR*, 2011.

[16] A. Hyvarinen and P. O. Hoyer. A two-layer sparse coding model learns simple and complex cell receptive fields and topography from natural images. *Vision Research*, 41(18):2413–23, 2001.

[17] P. Jain, B. Kulis, and K. Grauman. Fast image search for learned metrics. In *CVPR*, 2008.

[18] J.Wang, J. Yang, K. Yu, F. Lv, T. Huang, and Y. Gong. Locality-constrained linear coding for image classification. In *CVPR*, 2010.

[19] J.Wu and J. Rehg. Beyond the euclidean distance: Creating effective visual codebooks using the histogram intersection kernel. In *ICCV*, 2009.

[20] Y. Ke and R. Sukthankar. Pca-sift: A more distinctive representation for local image descriptors. In *CVPR*, 2004.

[21] J. Kim and G. K. Asymmetric region-to-image matching for comparing images with generic object categories. In *CVPR*, 2010.

[22] S. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *CVPR*, 2006.

[23] L. J. Li and L. Fei-Fei. What, where and who? classifying events by scene and object recognition. In *ICCV*, 2007.

[24] Z. Lin, A. Ganesh, J. Wright, L. Wu, M. Chen, and Y. Ma. Fast convex optimization algorithms for exact recovery of a corrupted low-rank matrix. In *Technical Report UILU-ENG-09-2214, UIUC*, August 2009.

[25] L. Liu, L. Wang, and X. Liu. In defense of softassignment coding. In *ICCV*, 2011.

[26] S. Liu, J. Feng, Z. Song, T. Zhang, H. Lu, C. Xu, and S. Yan. Hi, magic closet, tell me what to wear! In *ACM Multimedia*, 2012.

[27] D. G. Lowe. Distinctive image features from scaleinvariant keypoints. In *IJCV*, 2004.

[28] A. Oliva and A. Torralba. Modeling the shape of the scene: A holistic representation of the spatial envelope. *IJCV*, 42(1):145–175, 2001.

[29] Y. Peng, A. Ganesh, J. Wright, W. Xu, and Y. Ma. RASL: Robust Alignment by Sparse and Low-rank Decomposition for Linearly Correlated Images. *TPAMI (to appear)*, 2011.

[30] F. Sadeghi and M. F. Tappen. Latent pyramidal regions for recognizing scenes. In *ECCV*, 2012.

[31] A. Shabou and H. Le-Borgne. Locality-constrained and spatially regularized coding for scene categorization. In *CVPR*, 2012.

[32] S. Todorovic and N. Ahuja. Learning subcategory relevances for category recognition. In *CVPR*, 2008.

[33] J. C. van Gemert, J.-M. Geusebroek, C. J. Veenman, and A. W. M. Smeulders. Kernel codebooks for scene categorization. In *ECCV*, 2008.

[34] A. Vedaldi, V. Gulshan, M. Varma, and A. Zisserman. Multiple kernels for object detection. In *ICCV*, 2009.

[35] J. Yang, Y. Li, Y. Tian, L. Duan, and W. Gao. Group-sensitive multiple kernel learning for object categorization. In *ICCV*, 2009.

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

[37] K. Yu, T. Zhang, and Y. Gong. Nonlinear learning using local coordinate coding. In *NIPS*, 2009.

[38] H. Zhang, A. C. Berg, M. Maire, and J. Malik. Svm-knn: Discriminative nearest neighbor classification for visual category recognition. In *CVPR*, 2006.

[39] T. Zhang, B. Ghanem, S. Liu, and N. Ahuja. Low-rank sparse learning for robust visual tracking. In *European Conference on Computer Vision*, pages 1–14, 2012.

[40] T. Zhang, B. Ghanem, S. Liu, and N. Ahuja. Robust visual tracking via structured multi-task sparse learning. *International Journal of Computer Vision*, 101(2):367–383, 2013.