Linear Spatial Pyramid Matching Using Sparse Coding for Image Classification

Jianchao Yang[†], Kai Yu[‡], Yihong Gong[‡], Thomas Huang[†]
[†]Beckman Institute, University of Illinois at Urbana-Champaign
[‡]NEC Laboratories America, Cupertino, CA 95014, USA
[†]{jyang29, huang}@ifp.uiuc.edu, [‡]{kyu, ygong}@sv.nec-lab.com

Abstract

Recently SVMs using spatial pyramid matching (SPM) kernel have been highly successful in image classification. Despite its popularity, these nonlinear SVMs have a complexity $O(n^2 \sim n^3)$ in training and $O(n)$ in testing, where n is the training size, implying that it is nontrivial to scale-up the algorithms to handle more than thousands of training images. In this paper we develop an extension of the SPM method, by generalizing vector quantization to sparse coding followed by multi-scale spatial max pooling, and propose a linear SPM kernel based on SIFT sparse codes. This new approach remarkably reduces the complexity of SVMs to $O(n)$ in training and a constant in testing. In a number of image categorization experiments, we find that, in terms of classification accuracy, the suggested linear SPM based on sparse coding of SIFT descriptors always significantly outperforms the linear SPM kernel on histograms, and is even better than the nonlinear SPM kernels, leading to state-of-the-art performance on several benchmarks by using a single type of descriptors.

1. Introduction

In recent years the *bag-of-features* (BoF) model has been extremely popular in image categorization. The method treats an image as a collection of unordered appearance descriptors extracted from local patches, quantizes them into discrete "visual words", and then computes a compact histogram representation for semantic image classification, e.g. object recognition or scene categorization.

The BoF approach discards the spatial order of local descriptors, which severely limits the descriptive power of the image representation. By overcoming this problem, one particular extension of the BoF model, called *spatial pyramid matching* (SPM) [12], has made a remarkable success on a range of image classification benchmarks like Caltech-101 [14] and Caltech-256 [8], and was the major component of the state-of-the-art systems, e.g., [2]. The method partitions an image into $2^l \times 2^l$ segments in different scales $l = 0, 1, 2$, computes the BoF histogram within each of the 21 segments, and finally concatenates all the histograms to form a vector representation of the image. In case where only the scale $l = 0$ is used, SPM reduces to BoF.

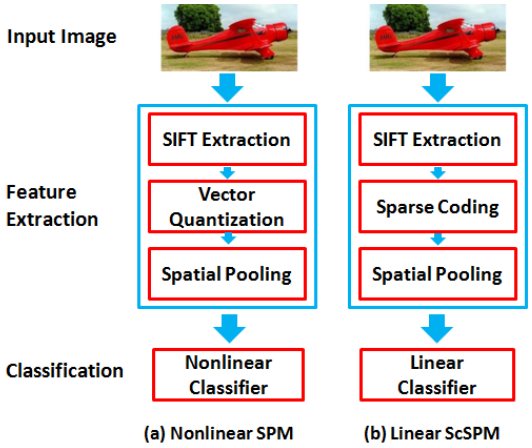

Figure 1. Schematic comparison of the original nonlinear SPM with our proposed linear SPM based on sparse coding (ScSPM). The underlying spatial pooling function for nonlinear SPM is *averaging*, while the spatial pooling function in ScSPM is *max pooling*.

People have empirically found that, in order to obtain good performances, both BoF and SPM must be applied together with a particular type of nonlinear Mercer kernels, e.g. the *intersection kernel* or the *Chi-square kernel*. Accordingly, the nonlinear SVM has to pay a computational complexity $O(n^3)$ and a memory complexity $O(n^2)$ in the training phase, where n is the training size. Furthermore, since the number of support vectors grows linearly with n, the computational complexity in testing is $O(n)$. This scalability implies a severe limitation — it is nontrivial to apply them to real-world applications, whose training size is typically far beyond thousands.

In this paper, we propose an extension of the SPM approach, which computes a spatial-pyramid image representation based on *sparse codes* (SC) of SIFT features, instead of the K-means vector quantization (VQ) in the traditional SPM. The approach is naturally derived by relaxing the restrictive cardinality constraint of VQ. Furthermore, unlike the original SPM that performs spatial pooling by computing histograms, our approach, called ScSPM, uses *max* spatial pooling that is more robust to local spatial translations and more biological plausible [24]. The new image representation captures more salient properties of visual patterns, and turns out to work surprisingly well with *linear* classifiers. Our approach using simple linear SVMs dramatically reduces the training complexity to $O(n)$, and obtains a constant complexity in testing, while still achieving an even better classification accuracy in comparison with the traditional nonlinear SPM approach. Schematic comparison between the original SPM with ScSPM is shown in Fig. 1.

The rest of the paper is organized as follows. In Sec. 2 we will talk about some related works. Sec. 3 presents the framework of our proposed algorithm and we give our efficient implementation in Sec. 4, followed by experiment results in Sec. 5. Finally, Sec. 6 concludes our paper.

2. Related Work

Over the years many works have been done to improve the traditional BoF model, such as generative methods in [7, 21, 3, 1] for modeling the co-occurrence of the codewords or descriptors, discriminative codebook learning in [10, 5, 19, 27] instead of standard unsupervised K-means clustering, and spatial pyramid matching kernel (SPM) [12] for modeling the spatial layout of the local features, all bringing promising progress. Among these extensions, motivated by Grauman and Darrell's pyramid matching in the feature space, the SPM proposed by Lazebnik *et al.* is particular successful.

As being easy and simple to construct, the SPM kernel turns out to be highly effective in practice. It contributes as the major component to the state-of-the-art systems, e.g., [2], and the systems of the top performers in PASCAL Challenge 2008 [6]. Despite of such a popularity, SPM has to run together with nonlinear kernels, such as the intersection kernel and the Chi-square kernel, in order to achieve a good performance, which requires intensive computation and a large storage. Realizing this, Anna Bosch *et al.* [2] used randomized trees instead of SVMs for faster training and testing. Most recently, Maji *et al.* [16] showed that one can build histogram intersection kernel SVMs much efficiently. However, the efficiency comes only for pre-trained nonlinear SVMs. In real applications which involves more than tens of thousands of training examples, linear kernel SVMs are far more favored as they enjoy both much faster training and testing speeds, with significantly less memory requirements compared to nonlinear kernels. Therefore, our proposed linear SPM using SIFT sparse codes is very promising in real applications.

Sparse modeling of image patches has been successfully applied to tasks such as image and video denoising, inpainting, demosaicing, super-resolution[5, 17, 26] and segmentation [18]. There are already some works devoting to image categorization through sparse coding on raw image patches [23, 22]. However, their performances are still behind the state-of-the-art achieved by [12, 1, 9] on public benchmarks. Our approach differs from them at using sparse coding on appearance descriptors like SIFT features, and the development of the whole system that achieves *state-of-the-art* performances on several benchmarks.

3. Linear SPM Using SIFT Sparse Codes

3.1. Encoding SIFT: From VQ to SC

Let $\mathbf{X}$ be a set of SIFT appearance descriptors in a D-dimensional feature space, i.e. $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_M]^\top \in \mathbb{R}^{M\times D}$. The vector quantization (VQ) method applies the K-means clustering algorithm to solve the following problem

$$\min_{\mathbf{V}} \quad \sum_{m=1}^{M} \min_{k=1\ldots K} \|\mathbf{x}_m - \mathbf{v}_k\|^2 \tag{1}$$

where $\mathbf{V} = [\mathbf{v}_1, \ldots, \mathbf{v}_K]^\top$ are the K cluster centers to be found, called *codebook*, and $\|\cdot\|$ denotes the $L2$-norm of vectors. The optimization problem can be re-formulated into a matrix factorization problem with cluster membership indicators $\mathbf{U} = [\mathbf{u}_1, \ldots, \mathbf{u}_M]^\top$,

$$\begin{aligned} \min_{\mathbf{U},\mathbf{V}} \quad & \sum_{m=1}^{M} \|\mathbf{x}_m - \mathbf{u}_m\mathbf{V}\|^2 \\ \text{subject to} \quad & Card(\mathbf{u}_m) = 1, |\mathbf{u}_m| = 1, \mathbf{u}_m \succeq 0, \forall m \end{aligned} \tag{2}$$

where $Card(\mathbf{u}_m) = 1$ is a cardinality constraint, meaning that only one element of $\mathbf{u}_m$ is nonzero, $\mathbf{u}_m \succeq 0$ means that all the elements of $\mathbf{u}_m$ are nonnegative, and $|\mathbf{u}_m|$ is the $L1$-norm of $\mathbf{u}_m$, the summation of the absolute value of each element in $\mathbf{u}_m$. After the optimization, the index of the only nonzero element in $\mathbf{u}_m$ indicates which cluster the vector $\mathbf{x}_m$ belongs to. In the training phase of VQ, the optimization Eq. (2) is solved with respect to both $\mathbf{U}$ and $\mathbf{V}$. In the coding phase, the learned $\mathbf{V}$ will be applied for a new set of $\mathbf{X}$ and Eq. (2) will be solved with respect to $\mathbf{U}$ only.

The constraint $Card(\mathbf{u}_m) = 1$ may be too restrictive, giving rise to often a coarse reconstruction of $\mathbf{X}$. We can relax the constraint by instead putting a $L1$-norm regularization on $\mathbf{u}_m$, which enforces $\mathbf{u}_m$ to have a small number

of nonzero elements. Then the VQ formulation is turned into another problem known as *sparse coding* (SC):

$$\min_{\mathbf{U},\mathbf{V}} \quad \sum_{m=1}^{M} \|\mathbf{x}_m - \mathbf{u}_m\mathbf{V}\|^2 + \lambda|\mathbf{u}_m| \qquad (3)$$
$$\text{subject to} \quad \|\mathbf{v}_k\| \le 1, \quad \forall k = 1, 2, \ldots, K$$

where a unit $L2$-norm constraint on $\mathbf{v}_k$ is typically applied to avoid trivial solutions [1]. Normally, the codebook $\mathbf{V}$ is an *overcomplete* basis set, i.e. $K > D$. Note that we drop out the nonnegativity constraint $\mathbf{u}_m \succeq 0$ as well, because the sign of $\mathbf{u}_m$ is not essential — it can be easily absorbed by letting $\mathbf{V}^\top \leftarrow [\mathbf{V}^\top, -\mathbf{V}^\top]$ and $\mathbf{u}_m^\top \leftarrow [\mathbf{u}_{m+}^\top, -\mathbf{u}_{m-}^\top]$ so that the constraint can be trivially satisfied, where $\mathbf{u}_{m+} = \max(0, \mathbf{u}_m)$ and $\mathbf{u}_{m-} = \max(0, \mathbf{u}_m)$.

Similar to VQ, SC has a training phase and a coding phase. First, a descriptor set $\mathbf{X}$ from a random collection of image patches is used to solve Eq. (3) with respect to $\mathbf{U}$ and $\mathbf{V}$, where $\mathbf{V}$ is retained as the codebook; In the coding phase, for each image represented as a descriptor set $\mathbf{X}$, the SC codes are obtained by optimizing Eq. (3) with respect to $\mathbf{U}$ only.

We choose SC to derive image representations because it has a number of attractive properties. First, compared with the VQ coding, SC coding can achieve a much lower reconstruction error due to the less restrictive constraint; Second, sparsity allows the representation to be specialize, and to capture salient properties of images; Third, research in image statistics clearly reveals that image patches are sparse signals.

3.2. Linear SPM

For any image represented by a set of descriptors, we can compute a single feature vector based on *some statistics* of the descriptors' codes. For example, if $\mathbf{U}$ is obtained via Eq. (2), a popular choice is to compute the histogram

$$\mathbf{z} = \frac{1}{M}\sum_{m=1}^{M} \mathbf{u}_m \qquad (4)$$

The bag-of-words approach to image classification computes such a histogram $\mathbf{z}$ for each image I represented by an unordered set of local descriptors. In the more sophisticated SPM approach, the image's spatial pyramid histogram representation $\mathbf{z}$ is a concatenation of local histograms in various partitions of different scales. After normalization $\mathbf{z}$ can be seen as again a histogram. Let $\mathbf{z}_i$ denote the histogram representation for image I_i. For a binary image classification problem, an SVM aims to learn a decision function

$$f(\mathbf{z}) = \sum_{i=1}^{n} \alpha_i \kappa(\mathbf{z}, \mathbf{z}_i) + b \qquad (5)$$

[1] For example, the objective can be decreased by respectively dividing and multiplying $\mathbf{u}_m$ and $\mathbf{V}$ by a constant factor.

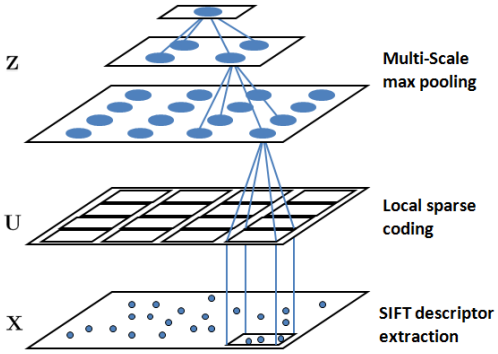

Figure 2. The illustration architecture of our algorithm based on sparse coding. Sparse coding measures the responses of each local descriptor to the dictionary's "visual elements". These responses are pooled across different spatial locations over different spatial scales.

where $\{(\mathbf{z}_i, y_i)\}_{i=1}^n$ is the training set, and $y_i \in \{-1, +1\}$ indicates labels. For a test image represented by $\mathbf{z}$, if $f(\mathbf{z}) > 0$ then the image is classified as positive, otherwise as negative. In theory $\kappa(\cdot,\cdot)$ can be any reasonable Mercer kernel function, but in practice the intersection kernel and Chi-square kernel have been found the most suitable on histogram representations. Our experiment shows that linear kernel on histograms leads to always substantially worse results, partially due to the high quantization error of VQ. However, using these two nonlinear kernels, the SVM has to pay a high training cost, i.e. $O(n^3)$ in computation, and $O(n^2)$ in storage (for the $n \times n$ kernel matrix). This means that it is difficult to scale up the algorithm to the case where n is more than tens of thousands. Furthermore, as the number of support vectors scales linearly to the training size, the testing cost is $O(n)$.

In this paper we advocate an approach of using *linear* SVMs based SC of SIFT. Let $\mathbf{U}$ be the result of applying the sparse coding Eq. (3) to a descriptor set $\mathbf{X}$, assuming the codebook $\mathbf{V}$ to be pre-learned and fixed, we compute the following image feature by a pre-chosen pooling function

$$\mathbf{z} = \mathcal{F}(\mathbf{U}), \qquad (6)$$

where the pooling function $\mathcal{F}$ is defined on each column of $\mathbf{U}$. Recall that each column of $\mathbf{U}$ corresponds to the responses of all the local descriptors to one specific item in dictionary $\mathbf{V}$. Therefore, different pooling functions construct different *image statistics*. For example, in 4, the underlying pooling function is defined as the *averaging* function, yielding the histogram feature. In this work, we defined the pooling function $\mathcal{F}$ as a *max* pooling function on the absolute sparse codes

$$z_j = \max\{|u_{1j}|, |u_{2j}|, ..., |u_{Mj}|\}, \qquad (7)$$

where z_j is the j-th element of $\mathbf{z}$, u_{ij} is the matrix element at i-th row and j-th column of $\mathbf{U}$, and M is the number of

local descriptors in the region. This max pooling procedure is well established by biophysical evidence in visual cortex (V1) [24] and is empirically justified by many algorithms applied to image categorization. In our case, we also find that max pooling outperforms other alternative pooling methods (see Sec. 5.5.4).

Similar to the construction of histograms in SPM, we do max pooling Eq. (7) on a spatial pyramid constructed for an image. By max pooling across different locations and over different spatial scales of the image, the pooled feature is more robust to local transformations than mean statistics in histogram. Fig. 2 illustrates the whole structure of our algorithm based on sparse coding. The pooled features from various locations and scales are then concatenated to form a spatial pyramid representation of the image.

Let image I_i be represented by $\mathbf{z}_i$, we use a simple linear SPM kernel

$$\kappa(\mathbf{z}_i, \mathbf{z}_j) = \mathbf{z}_i^\top \mathbf{z}_j = \sum_{l=0}^{2} \sum_{s=1}^{2^l} \sum_{t=1}^{2^l} \langle \mathbf{z}_i^l(s,t), \mathbf{z}_j^l(s,t) \rangle \quad (8)$$

where $\langle \mathbf{z}_i, \mathbf{z}_j \rangle = \mathbf{z}_i^\top \mathbf{z}_j$, and $\mathbf{z}_i^l(s,t)$ is the max pooling statistics of the descriptor sparse codes in the (s,t)-th segment of image I_i in the scale level l. Then the binary SVM decision function becomes

$$f(\mathbf{z}) = \left(\sum_{i=1}^{n} \alpha_i \mathbf{z}_i \right)^\top \mathbf{z} + b = \mathbf{w}^\top \mathbf{z} + b \quad (9)$$

In the literature, Eq. (5) is called the *dual formulation* of SVMs, while Eq. (9) is the *primal formulation*. As the major advantage of the linear kernel, now we can directly work in the primal, which means that the training cost is $O(n)$ in computation, and the testing cost for each image is even constant! In Sec. 4.2, we will describe our large-scale implementation for binary and multi-class linear SVMs.

Despite that the linear SPM kernel based on histograms leads to very poor performances, we find that the linear SPM kernel based on sparse coding statistics always achieves excellent classification accuracy. This success is largely due to three factors: (1) SC has much less quantization errors than VQ; (2) It is well known that image patches are sparse in nature, and thus sparse coding is particularly suitable for image data; (3) The computed statistics by max pooling are more salient and robust to local translations.

4. Implementation

4.1. Sparse Coding

The optimization problem Eq. (3) is convex in $\mathbf{V}$ (with $\mathbf{U}$ fixed) and convex in $\mathbf{U}$ (with $\mathbf{V}$ fixed), but not in both simultaneously. The conventional way for such a problem is to solve it iteratively by alternatingly optimizing over $\mathbf{V}$ or $\mathbf{U}$ while fixing the other. Fixing $\mathbf{V}$, the optimization can be solved by optimizing over each coefficient $\mathbf{u}_m$ individually:

$$\min_{\mathbf{u}_m} \|\mathbf{x}_m - \mathbf{u}_m \mathbf{V}\|_2^2 + \lambda |\mathbf{u}_m|. \quad (10)$$

This is essentially a linear regression problem with L_1 norm regularization on the coefficients, well known as Lasso in the Statistical literature. The optimization can be solved very efficiently by algorithms such as the recently proposed *feature-sign search* algorithm. [13]. Fixing $\mathbf{U}$, the problem reduces to a least square problem with quadratic constraints:

$$\begin{aligned} \min_{\mathbf{V}} \quad & \|\mathbf{X} - \mathbf{U}\mathbf{V}\|_F^2 \\ s.t. \quad & \|\mathbf{v}_k\| \le 1, \quad \forall k = 1, 2, ..., K. \end{aligned} \quad (11)$$

The optimization can be done efficiently by the Lagrange dual as used in [13].

In our experiments, we use $50,000$ SIFT descriptors extracted from random patches to train the codebook, by iterating the steps Eq. (10) and Eq. (11). Once we get the codebook $\mathbf{V}$ in this off-line training, we can do on-line sparse coding efficiently as in Eq. (10) on each descriptor of an image.

4.2. Multi-class Linear SVM

We introduce a simple implementation of linear SVMs that was used in our experiments. Given the training data $\{(\mathbf{z}_i, y_i)\}_{i=1}^n$, $y_i \in \mathcal{Y} = \{1, \ldots, L\}$, a linear SVM aims to learn L linear functions $\{\mathbf{w}_c^\top \mathbf{z} | c \in \mathcal{Y}\}$, such that, for a test datum $\mathbf{z}$, its class label is predicted by[2]

$$y = \max_{c \in \mathcal{Y}} \mathbf{w}_c^\top \mathbf{z} \quad (12)$$

We take a one-against-all strategy to train L binary linear SVMs, each solving the following unconstraint convex optimization problem

$$\min_{\mathbf{w}_c} \left\{ J(\mathbf{w}_c) = \|\mathbf{w}_c\|^2 + C \sum_{i=1}^{n} \ell\left(\mathbf{w}_c; y_i^c, \mathbf{z}_i\right) \right\} \quad (13)$$

where $y_i^c = 1$ if $y_i = c$, otherwise $y_i^c = -1$, and $\ell\left(\mathbf{w}_c; y_i^c, \mathbf{z}_i\right)$ is a hinge loss function. The standard hinge loss function is not differentiable everywhere, which hampers the use of gradient-based optimization methods. Here we adopt a differentiable quadratic hinge loss,

$$\ell\left(\mathbf{w}_c; y_i^c, \mathbf{z}_i\right) = \left[\max\left(0, \mathbf{w}_c^\top \mathbf{z} \cdot y_i^c - 1\right)\right]^2$$

such that the training can be easily done with simple gradient-based optimization methods. In our work we used

[2] The more general form of linear functions, i.e. $f(\mathbf{z}) = \mathbf{w}^\top \mathbf{z} + b$, can still be written as $f(\mathbf{z}) = \mathbf{w}^\top \mathbf{z}$ by adopting the reparameterization $\mathbf{w}^\top \leftarrow [\mathbf{w}^\top, b]$ and $\mathbf{z}^\top \leftarrow [\mathbf{z}^\top, 1]$.

LBFGS. Other choices like conjugate gradient are also applicable. The only implementation on our side is providing the cost $J(\mathbf{w})$ and the gradient $\partial J(\mathbf{w})/\partial \mathbf{w}$. The computation linearly scans over the training examples and thus has the *linear* complexity $O(n)$. In our experiment in Sec. 5.4, the SVM training on about $200,000$ examples with 5376-dimensional features was usually finished in 5 minutes.

5. Experiments and Results

In the experiments, we implemented and evaluated three classes of SPM methods on four diverse datasets: Caltech 101 [14], Caltech 256 [8], 15 Scenes [12], and TRECVID 2008 surveillance video. The three methods are

1. KSPM: the popular nonlinear kernel SPM that uses spatial-pyramid histograms and Chi-square kernels;

2. LSPM: the simple linear SPM that uses linear kernel on spatial-pyramid histograms;

3. ScSPM: the linear SPM that uses linear kernel on spatial-pyramid pooling of SIFT sparse codes,

Besides our own implementations, we also quote some results directly from the literature, especially those of KSPM from [12] and [8]. We note that sometimes we could not reproduce their results, largely due to subtle engineering details, e.g. the way of dealing with high-contrast and low-contrast patches. It thus makes more sense to compare our own implementations, since they were based on exactly the same set of descriptors.

Our implementations used a single descriptor type, the popular SIFT descriptor,[3] as in [12, 1, 9]. The SIFT descriptors extracted from 16×16 pixel patches were densely sampled from each image on a grid with stepsize 8 pixels. The images were all preprocessed into gray scale. To train the codebooks, we used standard K-means clustering for KSPM and LSPM, and the sparse coding scheme for our proposed ScSPM algorithm. For all the experiments except TRECVID 2008, we fixed the codebook size as 512 for LSPM and 1024 for ScSPM, to achieve optimal performances for both. For training the linear classifiers, we used our implemented SVM described in 4.2. The KSPM was trained using the LIBSVM [4] package.

Following the common benchmarking procedures, we repeat the experimental process by 10 times with different random selected training and testing images to obtain reliable results. The average of per-class recognition rates were recorded for each run. And we report our final results by the mean and standard deviation of the recognition rates.

[3] It is straightforward that the approach can be generalized to handle other descriptors and also multiple descriptors.

5.1. Caltech-101 Dataset

The Caltech-101 dataset contains 101 classes (including animals, vehicles, flowers, etc.) with high shape variability. The number of images per category varies from 31 to 800. Most images are medium resolution , i.e. about 300×300 pixels. We followed the common experiment setup for Caltech-101, training on 15 and 30 images per category and testing on the rest. Detailed comparison results are shown in Table 1. As shown, our sparse coding scheme outperforms linear SPM by more than 14 percent, and even outperform the nonlinear SPM [12] by a large margin (about 11 percent for 15 training and 9 percent for 30 training per category). One work needs to mention is the Kernel Codebooks [25], where the author assigned each descriptor into multiple bins instead of hard assignment. This scheme generally improves their baseline SPM by $5 \sim 6$ percent [4]. However, their method is still based on nonlinear kernels.

Table 1. Classification rate (%) comparison on Caltech-101.

Algorithms	15 training	30 training
Zhang et al. [28]	59.10 ± 0.60	66.20 ± 0.50
KSPM [12]	56.40	64.40 ± 0.80
NBNN [1]	65.00 ± 1.14	70.40
ML+CORR [9]	61.00	69.60
KC [25]	–	64.14 ± 1.18
KSPM	56.44 ± 0.78	63.99 ± 0.88
LSPM	53.23 ± 0.65	58.81 ± 1.51
ScSPM	$\mathbf{67.0 \pm 0.45}$	$\mathbf{73.2 \pm 0.54}$

5.2. Caltech-256 Dataset

The Caltech-256 dataset holds 29,780 images falling into 256 categories with much higher intra-class variability and higher object location variability compared with Caltech-101. Each category contains at least 80 images. We tried our algorithm on 15, 30, 45, and 60 training images per class respectively. The results are shown in Table 2. For all the cases, our ScSPM outperforms LSPM by more than 15 percent, and outperforms our own KSPM by more than 4 percent. In the cases of 45 and 60 training images per category, KSPM was not tried due to its very high computation cost for training.

5.3. 15 Scenes Categorization

We also tried our algorithm on the 15-Scenes dataset compiled by several researchers [20, 7, 12]. This dataset contains totally 4485 images falling into 15 categories, with the number of images each category ranging from 200 to 400. The 15 categories vary from living room and kitchen

[4] Because the codebook baseline scores are lower, the improved absolute performance obtained by the kernel codebook is not as high as may be obtained with a better baseline

Table 2. Classification rate (%) comparison on Caltech-256 dataset.

Algorithms	15 train	30 train	45 train	60 train
KSPM [8]	–	34.10	–	–
KC [25]	–	27.17 ± 0.46	–	–
KSPM	23.34 ± 0.42	29.51 ± 0.52	–	–
LSPM	13.20 ± 0.62	15.45 ± 0.37	16.37 ± 0.47	16.57 ± 1.01
ScSPM	$\mathbf{27.73 \pm 0.51}$	$\mathbf{34.02 \pm 0.35}$	$\mathbf{37.46 \pm 0.55}$	$\mathbf{40.14 \pm 0.91}$

to street and industrial. Following the same experiment procedure of Lazebnik et al. [12], we took 100 images per class for training and used the left for testing. The detailed comparison results are shown in Table 3. In this experiment, our implementation of kernel SPM was not able to reproduce the results reported in [12], probably due to the SIFT descriptor extraction and normalization process. Following our own baseline, the Linear ScSPM algorithm again achieves much better performance than KSPM and KC [25].

Table 3. Classification rate (%) comparison on 15 scenes.

Algorithms	Classification Rate
KSPM [12]	81.40 ± 0.50
KC [25]	76.67 ± 0.39
KSPM	76.73 ± 0.65
LSPM	65.32 ± 1.02
ScSPM	$\mathbf{80.28 \pm 0.93}$

5.4. TRECVID 2008 Surveillance Video

Figure 3. Examples of Events in TRECVID Surveillance Video

This time, we tried our algorithm on the large-scale data of 2008 TRECVID Surveillance Event Detection Evaluation, sponsored by National Institute of Standard and Technology (NIST). The data are 100 hours of surveillance videos, 10 hours each day, from London Gatwick International Airport. NIST defined 10 classes of events to detect, and provided 50 hours of annotated videos for training, as well as the other 50 hours videos for testing. The proposed algorithm of this paper was one of the main components in a system participating in 3 tasks of the evaluation, i.e. detecting *CellToEar*, *ObjectPut*, and *Pointing*, and being among the top performers. Some sample frames of these events are shown in Fig. 3. In addition to the event duration annotated by NIST, we manually marked the locations of persons performing the 3 events of interests.

The tasks are extremely challenging in two aspects: (1) The people subjects have a huge degree of variances in viewpoints and appearances, and are always in highly crowed and cluttered environments; (2) The detection system has to process 9 millions of 720×576 frames – the computation load is far beyond most of the research efforts known from the literature. To make the computation affordable, our system took a simple frame-based approach: first used a human detector to detect people subjects on each frame, and then applied classifiers on each detected region to further detect the events of interest. For each of the 3 events, we trained a binary classifier.

Table 4. AUC comparison on TRECVID 2008 surveillance video.

Algorithms	CellToEar	ObjectPut	Pointing
LSPM	0.688	0.714	0.744
ScSPM	**0.744**	**0.773**	**0.769**

Since the training videos were recorded in 5 different days, we used 5-fold cross validation to develop and evaluate our methods, where each fold corresponded to one day. In total, we got 2114, 2172, and 8725 positive examples of *CellToEar*, *ObjectPut*, and *Pointing*, respectively, and about 200,000 negative examples (only a small subset!) in the training set. Each example was a cropped image containing a detected human subject with the annotated event, resized into a 100×100 image. For each example, we extracted SIFT descriptors for every 16×16 patches on a grid of stepsize 8. The codebook sizes of both VQ and SC were set to be 256. Nonlinear SVM does not work on such a large-scale training set, therefore we only compared the two linear methods, ScSPM and LSPM. Due to the extremely unbalanced class distribution, we used ROC curves, as well as the AUC (area under ROC curve) scores to evaluate the accuracy. The average AUC results over 5 folds are shown

in Table 4. Typically, the SVM training on about $200,000$ examples with 5376-dimensional features was usually finished in 5 minutes.

5.5. Experiment Revisit

5.5.1 Patch Size

In our experiments, we only used one patch size to to extract SIFT descriptors, namely, 16×16 pixels as in SPM [12]. In NBNN[1], they used four patch scales to extract the descriptors in order to boost their performance. In our experiments, we didn't observe any substantial improvements by pooling over multiple patch scales, probably because max pooling over sparse codes can capture the salient properties of local regions that are irrelevant to the scale of local patches.

5.5.2 Codebook Size

We also investigated the effects of codebook sizes on these SPM algorithms. Intuitively, if the codebook size is too small, the histogram feature looses discriminant power; if the codebook size is too large, the histograms from the same class of images will never match. In Lazebnik et al.'s work, they used two codebook sizes 200 and 400 and reported that there was little difference. In our experiments on ScSPM and LSPM, we tried three sizes: 256, 512 and 1024. As shown in Table 5, the performance for LSPM increases initially and then decreases as the codebook size grows further. The performance for ScSPM continues to increase when the codebook size goes up to 1024.

Table 5. The effects of codebook size on ScSPM and LSPM respectively on Caltech 101 dataset.

	Codebook size	256	512	1024
30 train	ScSPM	68.26	71.20	**73.20**
	LSPM	57.42	**58.81**	58.56
15 train	ScSPM	61.97	63.23	**69.70**
	LSPM	51.84	**53.23**	51.74

5.5.3 Sparse Coding Parameter

There is one free parameter λ as in Eq. (10) we need to determine when we do sparse coding on each feature vector. λ enforces the sparsity of the solution; the bigger λ is, more sparse the solution will be. Empirically, we found that keeping the sparsity to be around 10% yields good results. For all our experiments, we simply fixed λ to be $0.3 \sim 0.4$ and the mean number of supports (non-zero coefficients) is around 10.

5.5.4 Comparison of Pooling Methods

We also studied two other straightforward pooling methods, namely, the square root of mean squared statistics (Sqrt) and the mean of absolute values (Abs), in comparison with *max* pooling. To be more precise, the other two pooling methods are defined as

$$\begin{aligned} Sqrt: \quad & z_j = \sqrt{\frac{1}{M}\sum_{i=1}^{M} u_{ij}^2} \\ Abs: \quad & z_j = \frac{1}{M}\sum_{i=1}^{M} |u_{ij}|, \end{aligned} \tag{14}$$

where the meanings of the notations are the same as in Eqn. 7. Experiments using three pooling methods on Caltech-101 for 30 training per categories and 15 Scenes for 100 training are listed in Table 6. As shown, *max* pooling produces the best performance, probably due to its robustness to local spatial variations.

Table 6. The performance comparison using different pooling methods on Caltech-101 and 15 Scenes for ScSPM.

	Sqrt	Abs	Max
Caltech	71.09 ± 1.47	66.68 ± 0.66	$\mathbf{73.2 \pm 0.54}$
Scenes	76.20 ± 0.77	73.92 ± 1.03	$\mathbf{80.4 \pm 0.45}$

5.5.5 Linear Kernel vs. Nonlinear Kernels

To justify the use of linear classifiers in our approach, we tried the popular *intersection kernel* and *Chi-square kernel* on our sparse coding features for comparison. We conducted the experiments on Caltech-101 (with 15 training examples) and 15 Scenes, and the results are shown in Table 7. As shown, our ScSPM based on linear kernel achieves a much better performance on both Caltech-101 and 15 Scenes compared to the nonlinear counterparts, not to mention that the nonlinear methods require much more computation. The compatibility of linear models with SIFT sparse codes is a very interesting phenomenon. One intuitive explanation is that, patterns with sparse features are more linearly separable, which is indeed the case for text classification.

Table 7. The performance comparison between linear and nonlinear kernels on ScSPM.

Dataset	Linear	Chi-Square	Intersection
Caltech	$\mathbf{67.0 \pm 0.45}$	60.7 ± 0.11	60.4 ± 0.98
Scene	$\mathbf{80.4 \pm 0.45}$	77.3 ± 0.75	77.7 ± 0.66

6. Conclusion and Future Work

In this paper we proposed a spatial pyramid matching approach based on SIFT sparse codes for image classification. The method uses selective sparse coding instead of traditional vector quantization to extract salient properties of appearance descriptors of local image patches. Further

more, instead of *averaging* pooling in the histogram, sparse coding enables us to operate local *max* pooling on multiple spatial scales to incorporate translation and scale invariance.

The most encouraging result of this paper is, the obtained image representation works surprisingly well with simple linear SVMs, which dramatically improves the scalability of training and the speed of testing, and even improves the classification accuracy. Our experiments on a variety of image classification tasks demonstrated the effectiveness of this approach. Since the nonlinear SPM based on vector quantization is very popular in top-performing image classification systems, we believe the suggested linear SPM will greatly improve *state-of-the-art* by allowing to use much larger sets of training data.

As an indication from our work, the sparse codes of SIFT features might serve as a better local appearance descriptor for general image processing tasks. Further research of this in empirical study and theoretical understanding is an interesting direction. Another issue is the efficiency of encoding. Currently encoding the SIFT descriptors of each Caltech image takes about 1 second in average. A recent work shows that sparse coding can be dramatically accelerated by using a feed-forward network [11]. It will be interesting to try such methods to make our approach faster. Moreover, the accuracy could be further improved by learning the codebook in a supervised fashion, as suggested by another recent work [15].

References

[1] O. Boiman, E. Shechtman, and M. Irani. In defense of nearest-neighbor based image classification. In *CVPR*, 2008. 2, 5, 7

[2] A. Bosch, A. Zisserman, and X. Munoz. Image classification using random forests and ferns. In *ICCV*, 2007. 1, 2

[3] A. Bosch, A. Zisserman, and X. Munoz. Scene classification using a hybrid generative/dicriminative approach. *TPAMI*, 2008. 2

[4] C.-C. Chang and C.-J. Lin. *LIBSVM: a library for support vector machines*, 2001. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`. 5

[5] M. Elad and M. Aharon. Image denoising via sparse and redundant representations over learned dictionaries. *IEEE Transaction on Image Processing*, 2006. 2

[6] M. Everingham, L. V. Gool, C. Williams, J. Winn, and A. Zisserman. The pascal visual object classes challenge 2008 (voc2008). In *ECCV Workshop*, 2008. 2

[7] L. Fei-Fei and P. Perona. A bayesian hierarchical model for learning natural scene categories. In *CVPR*, 2005. 2, 5

[8] Griffin, G. Holub, and P. AD. Perona. Caltech-256 object category dataset. Technical Report 7694, California Institute of Technology, 2007. 1, 5, 6

[9] P. Jain, B. Kullis, and K. Grauman. Fast image search for learned metrics. In *CVPR*, 2008. 2, 5

[10] F. Jurie and B. Triggs. Creating efficient codebooks for visual recognition. In *ICCV*, 2005. 2

[11] K. Kavukcuoglu, M. Ranzato, and Y. LeCun. Fast inference in sparse coding algorithms with applications to object recognition. Technical report, Computational and Biological Learning Lab, NYU, 2008. 8

[12] S. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *CVPR*, 2006. 1, 2, 5, 6, 7

[13] H. Lee, A. Battle, R. Raina, and A. Y. Ng. Efficient sparse coding algorithms. In *NIPS*, 2006. 4

[14] F.-F. Li, R. Fergus, and P. Perona. Learning generative visual models from few training examples:an incremental bayesian approach tested on 101 object categories. In *CVPR Workshop on Generative-Model Based Vision*, 2004. 1, 5

[15] J. Mairal, F. Bach, J. Ponce, G. Sapiro, and A. Zisserman. Supervised dictionary learning. In *NIPS*, 2009. 8

[16] S. Maji, A. C. Berg, and J. Malik. Classification using intersection kernel support vector machine is efficient. In *CVPR*, 2008. 2

[17] J. Malik, S. Belongie, T. Leung, and J. Shi. Sparse representation for color image restoration. *IEEE Transaction on Image Processing*, 2008. 2

[18] J. Mariral, F. Bach, J. Ponce, G. Sapiro, and A. Zisserman. Discriminative learned dictionaries for local image analysis. In *CVPR*, 2008. 2

[19] F. Moosmann, B. Triggs, and F. Jurie. Randomized clustering forests for building fast and discriminative visual vocabularies. In *NIPS*, 2007. 2

[20] A. Oliva and A. Torraba. Modeling the shape of the scene: A holistic representation of the spatial envelop. *IJCV*, 2001. 5

[21] P. Quelhas, F. Monay, J. Odobez, D. G.-P. T. Tuytelaars, and L. V. Gool. Modeling scenes with local descriptors and latent aspects. In *ICCV*, 2005. 2

[22] R. Raina, A. Battle, H. Lee, B. Packer, and A. Y. Ng. Self-taught learning: Transfer learning from unlabeled data. In *ICML*, 2007. 2

[23] M. Ranzato, F. Huang, Y. Boureau, and Y. LeCun. Unsupervised learning of invariant feature hierarchies with applications to object recognition. In *CVPR*, 2007. 2

[24] T. Serre, L. Wolf, and T. Poggio. Object recognition with features inspired by visual cortex. In *CVPR*, 2005. 2, 4

[25] J. C. van Gemert, J.-M. Geusebroek, C. J. Veenman, and A. W. M. Smeulders. Kernel codebooks for scene categorization. In *ECCV*, 2008. 5, 6

[26] J. Yang, J. Wright, T. Huang, and Y. Ma. Image super-resolution as sparse representation of raw image patches. In *CVPR*, 2008. 2

[27] L. Yang, R. Jin, R. Sukthankar, and F. Jurie. Unifying discriminative visual codebook generation with classifier training for object category recognition. In *CVPR*, 2008. 2

[28] H. Zhang, A. Berg, M. Maire, and J. Malik. Svm-knn: Discriminative nearest heighbor classification for visual category recognition. In *CVPR*, 2006. 5