Flexible Image Similarity Computation Using Hyper-Spatial Matching

Yu Zhang, Jianxin Wu, *Member, IEEE*, Jianfei Cai, *Senior Member, IEEE*, and Weiyao Lin

***Abstract*— Spatial pyramid matching (SPM) has been widely used to compute the similarity of two images in computer vision and image processing. While comparing images, SPM implicitly assumes that: in two images from the same category, similar objects will appear in similar locations. However, this is not always the case. In this paper, we propose hyper-spatial matching (HSM), a more flexible image similarity computing method, to alleviate the mis-matching problem in SPM. The match between corresponding regions, HSM considers the relationship of all spatial pairs in two images, which includes more meaningful match than SPM. We propose two learning strategies to learn SVM models with the proposed HSM kernel in image classification, which are hundreds of times faster than a general purpose SVM solver applied to the HSM kernel (in both training and testing). We compare HSM and SPM on several challenging benchmarks, and show that HSM is better than SPM in describing image similarity.**

***Index Terms*— Image similarity, spatial matching, fast SVM learning.**

I. Introduction and Related Research

SPATIAL pyramid matching (SPM) [1] has been widely used in computer vision problems to compute the similarity of two images, achieving state-of-the-art performance in various applications [2]–[9]. In SPM, an image is divided into N spatial non-overlapping or overlapping regions at different spatial levels, which form the spatial pyramid (SP). Then, each region is represented by a histogram of visual codewords using the bag-of-visual-words (BOV) model or other features, e.g., Fisher vector [10]. Finally, in classification or recognition, many methods compute the similarity of two images as the sum of the similarity of all corresponding regions using measures such as a linear [11] or non-linear additive kernel [12]–[14] κ, which can be solved efficiently by fast SVM classifiers, that is,

$$K_{SPM}(\boldsymbol{x}_i, \boldsymbol{x}_j) = \sum_{p=1}^{N} \kappa(\boldsymbol{x}_i^p, \boldsymbol{x}_j^p), \tag{1}$$

where $\boldsymbol{x}_i^p, \boldsymbol{x}_j^p \in \mathbb{R}^d$ are the feature vectors of the p-th spatial region in two images $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$, respectively.

SPM was originally proposed to compare corresponding spatial local regions in two images (that is, regions at the same relative spatial location in two images). It implicitly assumes that regions at the same spatial location in two images from the same category are similar. This implicit assumption is also inherited by most variants of SPM. In [3], each region in an image is assigned a learned weight (i.e., spatial saliency), and the similarity of two images is the sum of similarities of all weighted spatially corresponding regions. In [4], the authors used receptive fields, which are combinations of basic spatial regions, to provide more features in their image representation. Then, the image similarity is the sum of similarities of all spatially corresponding receptive fields. In [15], a specific spatial partition of images is learned for a given task. Similarly, [16] used different random partitions to split each image and find the optimal partition for each class to learn a classifier. In [17], the object location and its spatial subdivision are both learned in a latent structured SVM framework. In the face recognition area, different regions of faces are matched using the earth mover's distance in [18]. In [19], spatial regions of faces at multi-resolution are used to evaluate the face similarity, which leads to better face classification performance. More works focus on improving SPM by designing more powerful features for each region, see [5]–[9], [20]. In this paper, we focus on the matching of spatial regions, rather than the matching of interest points as in [21] and [22].

However, given two images in which similar objects appear in different spatial locations in them, SPM usually provides an inferior spatial matching result. In Fig. 1, two images of skyscrapers are divided into 4 non-overlapping regions. The sky and skyscrapers appear in different locations in these images, which make the spatially corresponding regions mismatch seriously. This phenomenon has also been observed in [23] and [24]. In [23], different spatial regions are mapped to a flexible spring lattice counting grid [25]. This method is suitable for scene images with shared parts. Researches in [24], [26], and [27] suggest to segment the foreground object from the background in each image and match them respectively. However, these approaches may not be suitable for scene images, which usually have no central objects.

Manuscript received August 28, 2013; revised March 23, 2014 and July 14, 2014; accepted July 25, 2014. Date of publication July 30, 2014; date of current version August 18, 2014. The work of J. Wu was supported by the National Natural Science Foundation of China under Grant 61321491. The work of W. Lin was supported by the National Natural Science Foundation of China under Grant 61001146. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Pascal Frossard.

Y. Zhang and J. Cai are with the School of Computer Engineering, Nanyang Technological University, Singapore 639798 (e-mail: yzhang4@e.ntu.edu.sg; asjfcai@ntu.edu.sg).

J. Wu is with the National Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210023, China (e-mail: wujx2001@nju.edu.cn).

W. Lin is with the Department of Electronic Engineering, Institute of Image Communication and Information Processing, Shanghai Jiao Tong University, Shanghai 200240, China (e-mail: wylin@sjtu.edu.cn).

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.2014.2344296

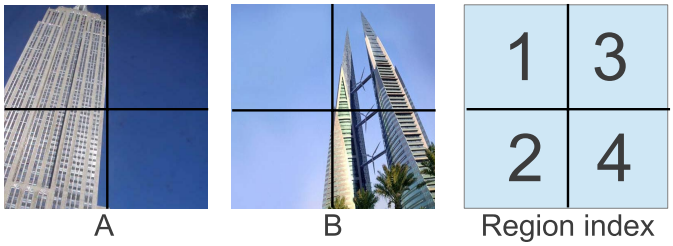


Fig. 1. Mis-matching of two images A and B in the same category (skyscraper) using SPM. The corresponding spatial regions in these two images mis-match seriously.

In this paper, we propose a novel hyper-spatial matching (HSM) kernel to alleviate the mis-matching problem in SPM. Specifically, we consider the pairwise relationship of *all* local regions in two images. The similarity between two images is defined as the sum of the weighted similarities of all spatial pairs, that is,

$$K_{HSM}(\boldsymbol{x}_i, \boldsymbol{x}_j) = \sum_{p=1}^{N}\sum_{q=1}^{N} R_{pq}\kappa(\boldsymbol{x}_i^p, \boldsymbol{x}_j^q), \tag{2}$$

where $R \in \mathbb{R}^N \times \mathbb{R}^N$ is a learned weight matrix encoding the relationships among different spatial regions.

HSM can include many variants of SPM, by assigning different values to the relationship matrix R. From Eq. 2, we can see that SPM is a special case of HSM, where R is an identity matrix. More importantly, with an appropriate R matrix, HSM can include meaningful matching between different spatial locations, e.g., between region 1 and 3, and region 2 and 4 of the two images in Fig. 1.

The value R_{pq} must represent the statistical relationship between the p-th and the q-th spatial region in all the (training) images. For example,

- A region in one image is usually similar to the spatially surrounding regions in images from the same category. It is thus intuitive to require that: R_{pq} should be relatively large if the p-th and the q-th regions are neighboring each other;
- If two regions are similar only in a few image pairs, then the corresponding weight in R should be small;
- When two images come from different categories, we should require that their HSM similarity is as small as possible; and,
- For a given problem or dataset, the learned R matrix should be stable. That is, if we use different subsets of images from the problem to learn several R matrices, they should be similar to each other.

In order to accommodate to these complicated constraints in a systematic manner, in this paper, the relationship matrix R is learned from the training images, by maximizing the HSM similarity of images from the same category, while at the same time minimizing the HSM similarity of images from different categories. In fact, after learning the R matrix in real datasets, we will visualize it and demonstrate that the above intuitions about the R matrix indeed hold. This fact also reflects the feasibility of the proposed HSM framework. Further discussions are provided in Section II-B (for showing the stability of R) and the experimental part of this paper (for validating other intuitions).

K_{HSM} is a Mercer kernel when R is a positive semidefinite (PSD) matrix and $\kappa(\cdot)$ is a Mercer kernel (cf. Section III-A). So it can be used within SVM directly for image classification tasks. Because additive kernels have shown excellent efficiency and efficacy in image classification tasks [13], [14], [28], we assume that the κ in Eq. 2 is additive (including the dot product kernel). However, it is easy to observe that HSM causes much more computations than SPM. A general purpose SVM solver like LIBSVM will be thousands of times slower than the recently designed additive kernel algorithms [13], [14], [28]. It is necessary to also devise a fast learning algorithm to apply HSM in large-scale problems.

Recent development of fast training algorithms for large scale problems include dual coordinate descent [11] and stochastic gradient descent [29] for the linear (dot product) kernel; intersection coordinate descent [12], piece-wise linear classifier [28], and nonlinear SVM [30] for the histogram intersection (HI) kernel; explicit data embedding [31], feature mapping [13] and power mean SVM (PmSVM) [14] for general additive kernels, etc. However, none of these fast methods can be directly applied to solve the HSM kernel learning problem.

In this paper, we propose two efficient strategies for HSM SVM learning and prediction: spectral linearization and gradient approximation. In spectral linearization, K_{HSM} learning is explicitly transformed into an approximate linear learning problem, and any linear SVM solver can be applied to solve it. In the gradient approximation strategy, the gradient of a dual HSM SVM objective is directly approximated by polynomial regression. Training and testing speed of both strategies for HSM are comparable to those of fast algorithms for linear and additive kernel SVM + SPM, and hundreds of times faster than general purpose SVM solvers + HSM. In addition, we propose a Hyper-spatial feature (HSF) to approximate HSM for high-dimensional features like Fisher vector (FV) [32] and VLAD [33] when a linear classifier is used. Thus, HSM can be readily applied to both large scale image classification problems, and to those problems requiring real-time testing speed.

Overall, the following contributions are presented in this paper:

- **Hyper-Spatial Matching**. HSM is proposed to alleviate the mis-matching in SPM, which is caused by considering only corresponding spatial regions in two images.

It evaluates all spatial pairs, which provides a better similarity metric than spatial pyramid matching. We also propose hyper-spatial features (HSF) if linear classifiers are preferred. HSF consistently improves classification results, with almost no increase in storage or CPU cost;

- **Efficient HSM SVM Learning and Prediction Algorithms**. The proposed strategies provide efficient and scalable learning and testing, in addition to high image classification accuracy. More importantly, they also provide explanations about why HSM provides better matching than SPM; and,
- **Accurate Classification Results**. HSM has higher classification accuracy rates than SPM on several challenging benchmark datasets, e.g., the SUN dataset [34].

II. Hyper-Spatial Matching

Before proposing the details of hyper-spatial matching, we first summarize the notations used in this paper:

- An image $\boldsymbol{x}$ has N regions $\boldsymbol{x} = (\boldsymbol{x}^1, \ldots, \boldsymbol{x}^N)$, and each region is represented by a d-dimensional feature vector $\boldsymbol{x}^i \in \mathbb{R}^d$;
- The training set $X = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$ contains n images. Each image is divided into N regions $\boldsymbol{x}_i = (\boldsymbol{x}_i^1, \ldots, \boldsymbol{x}_i^N)$, and has a class label $y_i \in \{1, 2, \ldots, N_c\}$, where N_c is the number of image categories; and,
- $x_i^{p,t}$ denotes the t-th dimension in the feature vector extracted from the p-th region in the i-th image in X, where $i = 1, \ldots, n$, $p = 1, \ldots, N$, and $t = 1, \ldots, d$.

We use an additive kernel κ to compare two regions throughout this paper. For more details of the additive kernels, the readers are referred to [13], [14], and [28].

A. Learn an Overall Relationship Matrix R

In hyper-spatial matching, the relationship matrix R plays a key role. We learn an overall relationship matrix R from the training samples of all categories, and require it to maximize the similarity of images from the same category, meanwhile to minimize the similarity of images from different categories. First, we define an $n \times n$ matrix D:

$$D_{ij} = \begin{cases} -\frac{1}{n^-} & \text{if } y_i = y_j \\ \frac{1}{n^+} & \text{otherwise,} \end{cases} \tag{3}$$

where n^- and n^+ are the number of same class pairs ($y_i = y_j$) and the number of cross class pair ($y_i \neq y_j$) in D, respectively. This matrix indicates whether images i and j belong to the same class or not. Since in a multiclass problem, the number of same class pairs is much smaller than that of pairs with different class labels, we normalize it with n^- and n^+ in Eq. 3, respectively.

For multi-label problems, D can be computed as:

$$D_{ij} = \begin{cases} -\frac{1}{n^-} & \text{if } \boldsymbol{y}_i \cap \boldsymbol{y}_j \neq \emptyset \\ \frac{1}{n^+} & \text{if } \boldsymbol{y}_i \cap \boldsymbol{y}_j = \emptyset, \end{cases} \tag{4}$$

where $\boldsymbol{y}_i$ and $\boldsymbol{y}_j$ are the label set related to the i-th and j-th instance, respectively.

Then, the learning process of R is formulated as:

$$\begin{aligned} \min_R \ & \sum_{i,j=1}^{n} \sum_{p,q=1}^{N} R_{pq} D_{ij} \kappa(\boldsymbol{x}_i^p, \boldsymbol{x}_j^q) + \gamma \|R\|_F^2 \\ \text{s.t. } & R \succeq 0, \ R_{pp} = 1, \quad \forall p. \end{aligned} \tag{5}$$

where $\|\cdot\|_F$ is the Frobenius norm of a matrix, $R \succeq 0$ means R is a positive semidefinite matrix, and $\gamma > 0$ is a parameter to balance the loss on the training set and the regularizer on R. Note that R is a symmetric matrix, based on the fact that the pairwise relationship is symmetric. Meanwhile, the values in its diagonal entries are set to 1, thus HSM can keep this same spatial correspondence as SPM. Besides, HSM can include more meaningful matching between different spatial locations if the off-diagonal values in R are non-zero.

Many existing kernels [35]–[38] can be viewed as special cases of HSM. For example, the kernels in [36]–[38] are used to compute the similarity of two bags (images) of local descriptors. When the bag sizes are the same, the sum-max kernel [36] is equivalent to HSM, by setting one specific column in each row of R to 1 (and 0 for all the rest). The kernels in [37], and [38] are equivalent to HSM by setting all values in R to 1.

The optimization in Eq. 5 is a standard semidefinite programming (SDP) problem, which can be efficiently solved. To see this, we change the order of summations in the objective function and define a new matrix $A \in \mathbb{R}^N \times \mathbb{R}^N$, where

$$A_{pq} = \sum_{i,j=1}^{n} D_{ij} \kappa(\boldsymbol{x}_i^p, \boldsymbol{x}_j^q). \tag{6}$$

This matrix encodes the average *affinity* or correlation of different regions, with the class labels y_i and y_j incorporated. Then, we can rewrite Eq. 5 in a more compact form, as:

$$\begin{aligned} \min_R \ & \operatorname{tr}(A^T R) + \gamma \operatorname{tr}(R^T R) \\ \text{s.t. } & R \succeq 0, \quad R_{pp} = 1, \quad \forall p, \end{aligned} \tag{7}$$

where $\operatorname{tr}(\cdot)$ is the trace of a matrix and $\operatorname{tr}(R^T R) = \|R\|_F^2$. Since N is small ($N = 31$ in this paper), it is a small scale SDP problem and can be solved in a few seconds.

In Fig. 2, we show an example of learned spatial relationships for normalized human faces. An R matrix is learned for all 7×7 regions with the size of 49×49. Relationships of three parts (eye, nose and mouth) to all 49 regions (which are three corresponding rows in R) are visualized according to R.[1] In each of the three similarity images, the dark red regions are the most similar to the query. We can see obvious symmetry between both eyes, and the self-similarity around the mouth and the nose.

B. Stability of the Affinity and Relationship Matrices

One important premise for the proposed hyper-spatial matching framework to work well for image similarity computation is that: *in a specific dataset, the relationship matrix R*

[1]This R is generated by replacing $\|R\|_F^2$ in Eq. 7 to $\|R\|_{\ell_1}$, in order to get a sparse R matrix for better visualization. In our experiments, the F-norm has slightly higher classification accuracy than the ℓ_1-norm, so the F-norm is adopted in Section IV and Eq. 7.

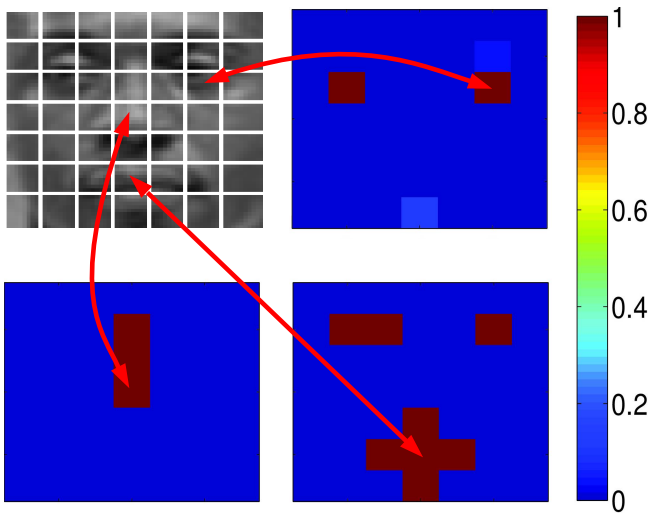


Fig. 2. Spatial relationship between different regions of human faces. This figure should be viewed in color.

learned from Eq. 7 should be stable. That is, if we use different images sampled from the same problem domain to calculate the affinity matrix A and to learn the relationship matrix R, we expect the learned matrices to be similar to each other.

Empirically, we observe that in our experiments both matrices A and R are in fact stable. We first sample five subsets from all available images of a specific dataset. Then, five versions of affinity matrices $A_1, \ldots, A_5$ are computed. Relationship matrices $R_1, \ldots, R_5$ are learned correspondingly. Next, we measure the variations of A or R by:

$$s_A = \frac{1}{5}\sum_{i=1}^{5}\frac{\|A_i - \bar{A}\|_F}{\|\bar{A}\|_F}, \quad s_R = \frac{1}{5}\sum_{i=1}^{5}\frac{\|R_i - \bar{R}\|_F}{\|\bar{R}\|_F}, \tag{8}$$

in which $\bar{A} = \frac{1}{5}\sum_{i=1}^{5} A_i$, $\bar{R} = \frac{1}{5}\sum_{i=1}^{5} R_i$, and $\|\cdot\|_F$ is the Frobenius norm.

The computed s_A and s_R values are usually very small. For example, when we randomly sample five subsets in the Scene 15 dataset (each with 10% examples from the training set), we find that $s_A = 0.057$ and $s_R = 0.065$ (s_R is computed by removing 1s in the diagonal of R_i and $\bar{R}$). Because both A and R are stable within a given problem, and computing the affinity matrix A is expensive, in this paper we simply use the same R for all experiments on one dataset. This relationship matrix is computed by sampling 10% examples from the training set.

C. Learn Independent Relationship Matrices R^c

An overall relationship matrix for all categories capture the global relationship between spatial regions. However, it may neglect the unique spatial arrangement of a specific category, e.g., two regions of one category are more similar than those of other categories, which must be highlighted by the relationship matrix. In this section, we learn one independent relationship matrix for each category to capture their unique spatial arrangements.

For each class $c \in \{1, 2, \ldots, N_c\}$, the relationship matrix R^c should maximize K_{HSM} of any two images in class c, and minimize the similarity of any two images with one from class c and the other from the rest classes. To achieve this goal, an $n \times n$ matrix D^c for class c (similar to Eq. 3) is first defined:

$$D^c_{ij} = \begin{cases} -\dfrac{1}{n^-} & y_i = y_j = c \\ \dfrac{1}{n^+} & (y_i \neq y_j) \wedge (y_i = c \vee y_j = c). \end{cases} \tag{9}$$

Finally, D^c is used to compute A^c and learn R^c similarly as Eq. 6 and 7, where A^c and R^c are the affinity and relationship matrices for class c, respectively.

III. Efficient HSM SVM Learning

HSM causes more computations than SPM in comparing two images: $O(N^2 d)$ vs. $O(Nd)$. Since it is not an additive kernel, recent developments of fast training algorithms for additive kernel SVM are not applicable either. In this section, we propose two novel fast learning strategies.

We choose SVM as the classifier and solve the dual SVM problem (without a bias term):

$$\begin{aligned} \min_{\boldsymbol{\alpha}} \quad & f(\boldsymbol{\alpha}) = \frac{1}{2}\boldsymbol{\alpha}^T Q\boldsymbol{\alpha} - \boldsymbol{e}^T\boldsymbol{\alpha} \\ \text{s.t.} \quad & 0 \leq \alpha_i \leq C, i = 1, \ldots, n, \end{aligned} \tag{10}$$

where $\boldsymbol{\alpha}$ consists of the Lagrange multipliers, $\boldsymbol{e}$ is a vector with all ones, and $Q_{ij} = y_i y_j K_{HSM}(\boldsymbol{x}_i, \boldsymbol{x}_j)$. Then, the classifier can be acquired as: $\boldsymbol{w} = \sum_{i=1}^{n} \alpha_i y_i \xi(\boldsymbol{x}_i)$, where ξ is an implicit mapping from the original input space to the feature space of K_{HSM}.

In the next two subsections, we propose two strategies to solve the problem (10): a spectral linearization of the problem (in Section III-A) and by directly approximating the gradient in SVM learning (in Section III-B).

A. The Spectral Linearization Strategy

The first strategy we propose to solve Eq. 10 is to linearize the non-linear HSM kernel approximately, using a spectral decomposition of R and the Fourier sampling technique in [13]. Any fast linear SVM solver can solve the converted approximate linear problem efficiently.

Since the additive kernel κ is a positive definite one, there exists a mapping ϕ that guarantees $\kappa(\boldsymbol{x}, \boldsymbol{y}) = \phi(\boldsymbol{x})^T\phi(\boldsymbol{y})$ for any examples (images) $\boldsymbol{x}$ and $\boldsymbol{y}$. With some abuse of notation, we define a 'matrix' $K_{\boldsymbol{x}}$ for any example $\boldsymbol{x}$, which is simply a reshaped version of $\phi(\boldsymbol{x})$:

$$K_{\boldsymbol{x}} = \begin{pmatrix} \phi(\boldsymbol{x}^1)^T \\ \vdots \\ \phi(\boldsymbol{x}^N)^T \end{pmatrix}. \tag{11}$$

Then, we define $K = K_{\boldsymbol{x}} K_{\boldsymbol{y}}^T$, in which $K_{\boldsymbol{y}}$ is defined similar to Eq. 11. So,

$$\begin{aligned} K_{HSM}(\boldsymbol{x}, \boldsymbol{y}) &= \sum_{p=1}^{N}\sum_{q=1}^{N} R_{pq}\phi(\boldsymbol{x}^p)^T\phi(\boldsymbol{y}^q) \\ &= \sum_{p=1}^{N}\sum_{q=1}^{N} R_{pq}K_{pq} = \mathrm{tr}(R^T K). \end{aligned} \tag{12}$$

where $K_{pq} = \kappa(\boldsymbol{x}^p, \boldsymbol{y}^q) = \phi(\boldsymbol{x}^p)^T\phi(\boldsymbol{y}^q)$. Note that K is dependent on $\boldsymbol{x}$ and $\boldsymbol{y}$, although we did not make this dependency explicit in the notation.

Because R is a real symmetric and positive semidefinite matrix, it has a spectral decomposition as:

$$R = \sum_{p=1}^{N} \lambda_p \boldsymbol{u}_p \boldsymbol{u}_p^T, \tag{13}$$

where $\lambda_p \geq 0$ and $\boldsymbol{u}_p \in \mathbb{R}^N$ are the eigenvalues and corresponding eigenvectors of R, respectively. With the spectral decomposition, a linearization of the HSM kernel is not difficult to figure out:

$$\begin{aligned} K_{HSM}(\boldsymbol{x}, \boldsymbol{y}) &= \mathrm{tr}(R^T K) \\ &= \mathrm{tr}\left(\sum_{p=1}^{N} \lambda_p \boldsymbol{u}_p \boldsymbol{u}_p^T K_{\boldsymbol{x}} K_{\boldsymbol{y}}^T\right) \\ &= \sum_{p=1}^{N} \lambda_p \mathrm{tr}\left(\boldsymbol{u}_p^T K_{\boldsymbol{x}} K_{\boldsymbol{y}}^T \boldsymbol{u}_p\right) \\ &= \sum_{p=1}^{N} \left(\sqrt{\lambda_p} K_{\boldsymbol{x}}^T \boldsymbol{u}_p\right)^T \left(\sqrt{\lambda_p} K_{\boldsymbol{y}}^T \boldsymbol{u}_p\right), \end{aligned} \tag{14}$$

in which we used the fact that $\mathrm{tr}(AB) = \mathrm{tr}(BA)$ and $\mathrm{tr}(x) = x$ when $x \in \mathbb{R}$. Then, the following mapping Ψ provides a linearization for HSM, and $K_{HSM}(\boldsymbol{x}, \boldsymbol{y}) = \Psi(\boldsymbol{x})^T\Psi(\boldsymbol{y})$ for all $\boldsymbol{x}$ and $\boldsymbol{y}$:

$$\Psi : \boldsymbol{x} = (\boldsymbol{x}^1, \ldots, \boldsymbol{x}^N) \mapsto (\sqrt{\lambda_1} K_{\boldsymbol{x}}^T \boldsymbol{u}_1, \ldots, \sqrt{\lambda_N} K_{\boldsymbol{x}}^T \boldsymbol{u}_N). \tag{15}$$

For many additive kernels κ, the mapping ϕ could be high or even infinite-dimensional (or does not have an explicit form), rendering Eq. 15 impractical. However, we can use the random Fourier sampling technique in [13] to reach an approximate linearization for additive kernels. An additive kernel κ can be approximated by $\kappa(\boldsymbol{x}, \boldsymbol{y}) \approx \hat{\phi}(\boldsymbol{x})^T\hat{\phi}(\boldsymbol{y})$. The feature mapping $\hat{\phi}$ transforms any scalar value x into $2m+1$ dimensions, and $\hat{\phi}(\boldsymbol{x}) \in \mathbb{R}^{(2m+1)d}$ if $\boldsymbol{x} \in \mathbb{R}^d$. Details of the the Fourier sampling technique and examples of closed form feature mapping $\hat{\phi}$ for some commonly used additive kernels can be found in [13].

Replacing ϕ with $\hat{\phi}$ in Eq. 11 and Eq. 15, for any $\boldsymbol{x}$ in an HSM setting, $\hat{K}_{\boldsymbol{x}}$ is now a $N \times (2m+1)d$ matrix, $\hat{K}_{\boldsymbol{x}}^T \boldsymbol{u}_p$ is $(2m+1)d$ dimensional, and the approximate HSM linearization $\hat{\Psi}$ is a $(2m+1)Nd$ dimensional vector.

Then, any fast linear SVM solver can be used to solve the linearized problem. This fact shows that K_{HSM} is a Mercer kernel when R is a PSD matrix. The complete learning process is shown in Algorithm 1.

Before presenting the next learning strategy, we end this subsection with a technical note. Eq. 15 shows that $R \succeq 0$ is a sufficient condition for HSM to be a valid Mercer kernel. In fact, $R \succeq 0$ is also a necessary condition. If R is not positive semidefinite, then $K_{HSM}(\boldsymbol{x}, \boldsymbol{x}) < 0$ may hold, which cannot happen in a Mercer kernel. One such example is when $R = \left(\begin{smallmatrix} 1 & 2 \\ 2 & 1 \end{smallmatrix}\right)$, $N = 2$, $d = 1$, $\boldsymbol{x} = (1, -2)$, and $\kappa(x, y) = xy$, we have $K_{HSM}(\boldsymbol{x}, \boldsymbol{x}) = -3$.

Algorithm 1 HSM SVM Learning by Spectral Linearization

1: Convert dataset: for any training example $\boldsymbol{x}$ (replacing ϕ with $\hat{\phi}$ in relevant equations):
   1) Compute $\hat{K}_{\boldsymbol{x}}$ using Eq. 11;
   2) Convert $\boldsymbol{x}$ to $\hat{\Psi}(\boldsymbol{x})$ using Eq. 15.
2: Train a linear SVM with the new dataset $(\hat{\Psi}(\boldsymbol{x}_i), y_i)$, $i = 1, \ldots, n$. The converted examples and the trained linear classifier $\boldsymbol{w}$ are $(2m+1)Nd$ dimensional.
3: **Classification**: for a new test image $\boldsymbol{x} = (\boldsymbol{x}^1, \ldots, \boldsymbol{x}^N)$,
   1) Convert $\boldsymbol{x}$ to $\hat{\Psi}(\boldsymbol{x})$ using the above conversion procedure;
   2) Output: $\mathrm{sgn}\left(\boldsymbol{w}^T\hat{\Psi}(\boldsymbol{x})\right)$.

Algorithm 2 The Coordinate Descent Method [39]

1: **Input**: Given the Lagrange multipliers $\boldsymbol{\alpha}$ and the corresponding $\boldsymbol{w} = \sum_{i=1}^{n} \alpha_i y_i \xi(\boldsymbol{x}_i)$.
2: Compute $Q_{ii} = \|\xi(\boldsymbol{x}_i)\|_{\ell_2}^2, i = 1, \ldots, n$.
3: **while** $\boldsymbol{\alpha}$ is not optimal **do**
4: **for** $i = 1, \ldots, n$ **do**
5: Compute $G = y_i \boldsymbol{w}^T \xi(\boldsymbol{x}_i) - 1$
6: $\overline{\alpha}_i \leftarrow \alpha_i$
7: $\alpha_i \leftarrow \min(\max(\alpha_i - G/Q_{ii}, 0), C)$
8: $\boldsymbol{w} \leftarrow \boldsymbol{w} + y_i(\alpha_i - \overline{\alpha}_i)\xi(\boldsymbol{x}_i)$
9: **end for**
10: **end while**

B. The Gradient Approximation Strategy

The second strategy we propose directly approximates the gradient in coordinate descent SVM learning. This strategy is better than the spectral linearization strategy in practice: SVM training is faster and requires less memory; meanwhile it has a slightly higher classification accuracy (cf. Section IV).

In order to efficiently solve the problem (10), we choose the dual coordinate descent method [39], which is shown in Algorithm 2. We approximate the gradient G (line 5 in Algorithm 2) using the gradient approximation method in [14].

First, we substitute K_{HSM} into G:

$$G = y_i \sum_{p=1}^{N} \sum_{t=1}^{d} \sum_{q=1}^{N} R_{pq} \sum_{j=1}^{n} \alpha_j y_j \kappa(x_i^{p,t}, x_j^{q,t}) - 1. \tag{16}$$

It needs $O(nN^2d)$ steps to compute, which is very expensive. In Eq. 16, the essential part is $\sum_{j=1}^{n} \alpha_j y_j \kappa(x, x_j^{q,t})$ for any scalar value x. We define this summation as a function $g_t^q(x)$.

Then, we use an m' degree polynomial regression model to approximate it (following [14]):

$$g_t^q(x) = \sum_{j=1}^{n} \alpha_j y_j \kappa(x, x_j^{q,t}) \approx \sum_{s=0}^{m'} a_s^{q,t} x^s, \tag{17}$$

where $a_s^{q,t}$ is the polynomial regression parameter for the t-th dimension in the q-th region, and x^s is x raised to the s-th power.

Algorithm 3 HSM SVM Learning by Gradient Approximation

$\boldsymbol{\alpha} \leftarrow 0$
$b_s^{p,t} \leftarrow 0$, $p = 1, \ldots, N$, $t = 1, \ldots, d$, $s = 0, 1, \ldots, m'$
$Q_{ii} \leftarrow K_{HSM}(\boldsymbol{x}_i, \boldsymbol{x}_i), i = 1, \ldots, n.$
while $\boldsymbol{\alpha}$ is not optimal **do**
  for $i = 1, \ldots, n$ **do**
    Compute G using Eq. 20.
    $\overline{\alpha}_i \leftarrow \alpha_i$
    $\alpha_i \leftarrow \min(\max(\alpha_i - g/Q_{ii}, 0), C)$
    $\boldsymbol{b}^{p,t} \leftarrow \boldsymbol{b}^{p,t} +$
      $(\alpha_i - \overline{\alpha}_i) y_i X^{-1} \sum_{q=1}^{N} R_{pq} \kappa(\boldsymbol{c}, x_i^{q,t})$, $\forall p, t$.
  end for
end while
Classification: For a new test image $\boldsymbol{x} = (\boldsymbol{x}^1, \ldots, \boldsymbol{x}^N)$ divided into N regions, the classification result is:
$\operatorname{sgn}\left(\sum_{p=1}^{N} \sum_{t=1}^{d} (\sum_{s=0}^{m'} b_s^{p,t} (x^{p,t})^s)\right)$.

Next, we need to learn the $m' + 1$ parameters $a_s^{q,t}$ in the model. We choose $m' + 1$ values $\boldsymbol{c} = (c_0, \ldots, c_{m'})$ between [0 1] and calculate $g_t^q(c_i)$, $i = 0, \ldots, m'$. The optimal parameters of $\boldsymbol{a}^{q,t} = (a_0^{q,t}, \ldots, a_{m'}^{q,t})$ equals [14]

$$\boldsymbol{a}^{q,t} = X^{-1} g_t^q(\boldsymbol{c}), \tag{18}$$

where X is a Vandermonde matrix with $X_{ij} = (c_i)^j, \forall i, j$, and $g_t^q(\boldsymbol{c}) = (g_t^q(c_0), \ldots, g_t^q(c_{m'}))$. To simplify the notations, we denote $b_s^{p,t} = \sum_{q=1}^{N} R_{pq} a_s^{q,t}$, and the vector form is now:

$$\boldsymbol{b}^{p,t} = \sum_{q=1}^{N} R_{pq} \boldsymbol{a}^{q,t}. \tag{19}$$

Finally, we have the gradient approximated as:

$$G = y_i \sum_{p=1}^{N} \sum_{t=1}^{d} \left(\sum_{s=0}^{m'} b_s^{p,t} (x_i^{p,t})^s \right) - 1. \tag{20}$$

Meanwhile, given a learned classifier $\boldsymbol{w}$ and a new testing image $\boldsymbol{x}$, the SVM decision value is:

$$\boldsymbol{w}^T \xi(\boldsymbol{x}) = \sum_{p=1}^{N} \sum_{t=1}^{d} \left(\sum_{s=0}^{m'} b_s^{p,t} (x^{p,t})^s \right). \tag{21}$$

In Eq. 20 and 21, m' is usually very small, e.g., $m' = 2$, so the complexity are greatly reduced from $O(nN^2d)$ to $O(Nd)$. Note that Nd is the length of the whole feature vector for an image in SPM. Thus, *HSM has the same testing complexity as a linear classifier*. The complete learning process of HSM SVM using the gradient approximation strategy is given in Algorithm 3.

Note that if we want to compare the two proposed HSM SVM strategies, we need to make $2m + 1$ (the number of dimensions in $\boldsymbol{w}$ corresponding to a single dimension in the input in Algorithm 1) and $m' + 1$ (in Algorithm 3) equal to each other. In this paper, we choose $m = 1$ and $m' = 2$ in our experiments.

Algorithm 4 Dot Product HSM SVM

$\boldsymbol{\alpha} \leftarrow 0$, $\hat{\boldsymbol{w}} \leftarrow 0$.
$Q_{ii} \leftarrow \sum_{p=1}^{N} \sum_{q=1}^{N} R_{pq} (\boldsymbol{x}_i^p)^T (\boldsymbol{x}_i^q), i = 1, \ldots, n.$
while $\boldsymbol{\alpha}$ is not optimal **do**
  for $i = 1, \ldots, n$ **do**
    Compute G using Eq. 22.
    $\overline{\alpha}_i \leftarrow \alpha_i$
    $\alpha_i \leftarrow \min(\max(\alpha_i - g/Q_{ii}, 0), C)$
    $\hat{w}^{p,t} \leftarrow \hat{w}^{p,t} + (\alpha_i - \overline{\alpha}_i) y_i \sum_{q=1}^{N} R_{pq} x_i^{q,t}$
  end for
end while
Classification: For a test image $\boldsymbol{x}$, output: $\operatorname{sgn}(\hat{\boldsymbol{w}}^T \boldsymbol{x})$.

C. Dot Product HSM SVM

Both proposed strategies deal with HSM with a base non-linear additive kernel. When the base kernel κ is a dot product kernel, we can directly derive an *exact* formulation for HSM SVM:

$$G = y_i \sum_{p=1}^{N} \sum_{t=1}^{d} x_i^{p,t} \left(\sum_{q=1}^{N} R_{pq} \sum_{j=1}^{n} \alpha_j y_j x_j^{q,t} \right) - 1, \tag{22}$$

and the decision value is:

$$\boldsymbol{w}^T \xi(\boldsymbol{x}) = \sum_{p=1}^{N} \sum_{t=1}^{d} x^{p,t} \left(\sum_{q=1}^{N} R_{pq} \sum_{j=1}^{n} \alpha_j y_j x_j^{q,t} \right). \tag{23}$$

By defining $\hat{\boldsymbol{w}}$, where $\hat{w}^{p,t} = \sum_{q=1}^{N} R_{pq} \sum_{j=1}^{n} \alpha_j y_j x_j^{q,t}$, we get $\boldsymbol{w}^T \xi(\boldsymbol{x}) = \hat{\boldsymbol{w}}^T \boldsymbol{x}$, which has the same complexity as a linear classifier. The learning process is shown in Algorithm 4.

D. The Decorrelation Effect of HSM

The fundamental motivation of HSM comes from the observation that different spatial regions are in fact interacting with each other; or, *correlated with each other*. Thus, we expect the matrix A that encodes affinity of regions (computed from Eq. 6) has many non-zero off-diagonal entries. These correlations make HSM highly non-linear.

However, as shown in the spectral linearization based strategy, HSM can be approximately linearized. Since the linearized version is an approximation of the original non-linear HSM problem, in the linearized problem, *different regions should not correlate with each other any more*. That is, we would expect *the relationship matrix R of HSM to decorrelate those spatial regions that are originally correlated.* This statement is empirically verified in Fig. 3.

The left image in Fig. 3 visualizes the affinity matrix A computed from the original dataset. It is obvious that although the diagonal entries have the largest absolute values, many off-diagonal entries have correlations that can not be ignored. However, if we compute the affinity matrix A again on the linearized dataset, the middle image demonstrates that most off-diagonal entries are close to 0. That is, after applying HSM to linearize the problem, different spatial regions now are *not* correlated with each other in most cases.

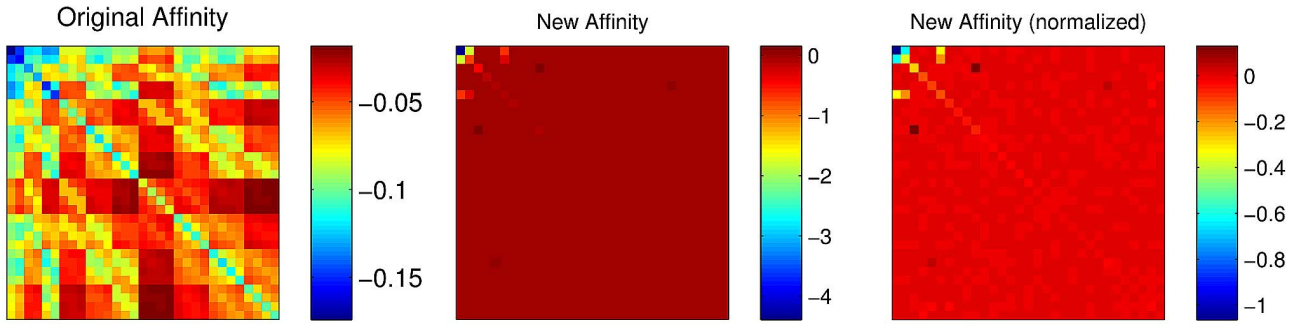


Fig. 3. Affinity matrix (computed on the Scene 15 problem [1]) of the original dataset; of the linearized dataset after applying HSM; and the eigenvalue normalized version after applying HSM, respectively. This figure should be viewed in color.

Two points deserve further attention. First, because in Eq. 15, we sort the eigenvalues as $\lambda_1 \geq \lambda_2 \geq \cdots$, the values in the middle image decreases from top to bottom and from left to right. If we remove the effect of eigenvalues (i.e., remove all λ_i from Eq. 15), the affinity matrix of the linearized dataset is shown in the right part of Fig. 3. It is obvious that the decorrelation effect still holds.

Second, there are a few remaining off-diagonal entries with relatively large values after linearization. However, all of them involve regions from the level 0 or level 1 in the spatial pyramid. Such regions cover the entire image (level 0) or a quarter of the entire image (level 1). Thus, it is possible that they still have remaining correlation with level 2 regions that are contained in them.

Overall, we believe that

- Correlations exist among regions at different spatial locations, which cannot be safely ignored; and,
- Through the learned relationship matrix R, HSM successfully decorrelates different spatial regions.

These observations may explain why HSM has an advantage over SPM in various image categorization tasks in our experiments (cf. Section IV).

E. SVM Learning With Independent Relationship Matrices

Until now, we only consider the HSM kernel with one overall relationship matrix. We want to emphasize that both strategies can readily use independent relationship matrices.

In the multi-class classification problem, we use the one-vs-rest strategy. During the learning phase, images of one class $c \in \{1, 2, \ldots, N_c\}$ are chosen as positive samples, and the rest images are treated as negative samples. A binary classifier is learned for this class c with R^c.

The choice to use multiple independent relationship matrices (rather than one overall relationship matrix) has influence to the two proposed strategies. For the gradient approximation strategy, N_c relationship matrices means that all training instances need to compute N_c times of Eq. 19. When N_c is large, the cost of this gradient transformation will be evident in the whole computation. However, Eq. 19 only influences the training stage. In the testing phase, the computational cost for an example remains the same as that using one overall relationship matrix. Similarly, the conversion step (Eq. 15) needs to be run N_c times for any example $\boldsymbol{x}$ in spectral linearization during the training stage, and the testing cost is similar as that of the gradient approximation strategy.

F. Hyper-Spatial Features

For very high dimensional visual representations such as Fisher vector [10] and VLAD [33], even Algorithm 4 could be too costly. Thus, we propose a simple way to append *hyper-spatial features* (HSF) to approximate HSM. Given an image with N regions, $\boldsymbol{x} = (\boldsymbol{x}^1, \ldots, \boldsymbol{x}^N)$, we append the following $\frac{N(N-1)}{2}$ values to the end of $\boldsymbol{x}$:

$$(\boldsymbol{x}^p)^T(\boldsymbol{x}^q), \quad 1 \leq p < q \leq N. \tag{24}$$

When using the linear (dot product) kernel on the expanded vectors, the above appended within-image spatial similarity values will approximate the HSM similarity.

Encoding spatial information is related to [40], which appends coordinates into local visual descriptors to include the spatial information. In contrast, HSF encodes the spatial relation between regions rather than among local descriptors. In other words, HSF values are additional useful contextual information in an image.

IV. Experimental Results

In this section, we empirically compare HSM with state-of-the-art methods. The two proposed strategies are abbreviated as "ga" and "sl", to denote gradient approximation and spectral linearization, respectively. We use the two most widely used additive kernels for κ: χ^2 and histogram intersection (HI). They are denoted as HSM-ga(sl)-χ^2(HI). We also denote HSM with linear kernel as HSM-linear. Besides, we evaluate the performance of HSM using the independent relationship matrices with the gradient approximation strategy, denoted as HSM$_{ind}$-ga-χ^2(HI). All other HSM results use an overall R.

Eq. 7 is solved by using the CVX [41] software package and $\gamma = 0.2$. In Eq. 10, $C = 0.01$ for gradient approximation with non-linear kernels, $C = 1$ for spectral linearization with non-linear kernels, and $C = 1$ for the linear kernel, respectively.[2] In multi-class classification tasks, we use the one-vs-rest strategy. An overview of HSM methods is listed in Table I.

[2]These values are default values for the PmSVM and LIBLINEAR software package, respectively.

TABLE I
OVERVIEW OF HSM METHODS IN THE EXPERIMENT

notation	κ	SVM solving method	R / R^c
HSM-ga-χ^2	χ^2	gradient approximation	R
HSM-ga-HI	HIK	gradient approximation	R
HSM-sl-χ^2	χ^2	spectral linearization	R
HSM-sl-HI	HIK	spectral linearization	R
HSM$_{ind}$-ga-χ^2	χ^2	gradient approximation	R^c
HSM$_{ind}$-ga-HI	HIK	gradient approximation	R^c
HSM-linear	linear	Algorithm 4	R
HSM-exact-χ^2	χ^2	precomputed kernel	R

In Algorithm 3, $X^{-1}\sum_{q=1}^{N} R_{pq}\kappa(\boldsymbol{c}, x_i^{q,t})$ in Line 9 is used frequently during the iterations, which can be precomputed and stored to save lots of computations in this process. It is the same for Line 8 of Algorithm 4. If the dataset is too large to fit into the memory, we have to compute it in every iteration, however, which is still more efficient than other available methods, e.g., LIBSVM [42]. In this paper, all datasets can fit into main memory, so we use the former strategy.

Experiments are organized as follows. First, we compare the proposed approximate learning strategies with exact HSM SVM. We use LIBSVM [42] to implement the exact HSM kernel K_{HSM}. In order to use the HSM kernel in LIBSVM, we precompute the kernel matrices for both training and testing purposes, which are used as the input of LIBSVM with the precomputed kernel option ("-t 4").

Second, we compare HSM with SPM using both proposed strategies and available fast classifiers with linear / nonlinear base kernels. Two fast methods with nonlinear kernels are used for SPM:

PmSVM [14] (SPM-ga). PmSVM includes a family of additive kernels $\kappa(x, y) = (\frac{x^p+y^p}{2})^{1/p}, x, y \in \mathbb{R}$. When $p = -1$, it is the χ^2 kernel, we denote it as SPM-ga-χ^2. When $p = -\infty$, it is the histogram intersection kernel $\kappa(x, y) = \min(x, y)$. In [14], the author uses $p = -8$ to approximate the HI kernel denoted by SPM-ga-HI. We set $C = 0.01$.

Feature Mapping (fm) [13] (SPM-sl). We map each dimension of the original feature vector into a 3D vector (that is, $m = 1$) and concatenate them altogether into a long vector. These mapped feature vectors are used to train a linear classifier with LIBLINEAR, where $C = 1$.

The fast linear classifier for SPM we adopted in our experiments is:

LIBLINEAR [11] (SPM-Linear). LIBLINEAR is a fast linear SVM solver using dual coordinate descent. We use its default parameter $C = 1$.

We evaluate the above methods on four datasets. They are:

Scene 15 [1]. It contains 15 scene classes. 100 images from each class are used for training the classifier and all the rest images are used for testing.

Indoor 67 [43]. The dataset contains 67 classes. 80 images of each class are used for training and 20 for testing.

SUN [34]. It has 397 classes. In each class, we use 50 training images and 50 testing images.

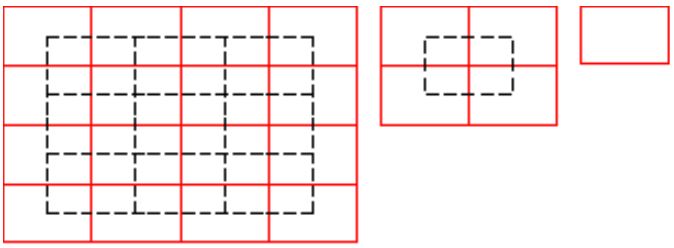


Fig. 4. Three level spatial pyramid of an image. The three figures show level 2, 1, and 0, respectively [9].

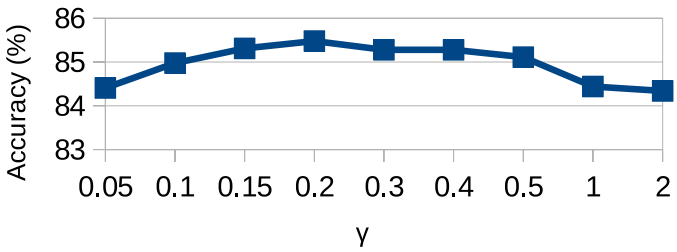


Fig. 5. Classification accuracy w.r.t. γ in Eq. 7 on Scene 15 using HSM-ga-χ^2.

VOC2007 [44]. It is a multi-label categorization problem. We use the provided training and validation set for training, and the testing set for testing.

A three level spatial pyramid in Fig. 4 is used for BOV, following the pyramid structure in [9]. The region numbers are 1, 5, and 25 at the three levels of the spatial pyramid, respectively. We extract SIFT descriptors for all the datasets. In the BOV process, different codeword numbers are used for different datasets in order to compare with existing works fairly. We also evaluate HSF + Fisher vector (FV) and HSF + VLAD. All feature vectors are subject to the power normalization [8] and ℓ_2 normalization for each region. Following the setup in [45], we use non-linear classifiers for BOV and linear classifier for FV.

We report the classification accuracy, the training time and the testing time of all methods in the experiments. All the results of each dataset (except VOC 2007) are averaged on 5 rounds by randomly splitting the training and testing sets. We also compare our methods with recently published results on these four datasets. Experiments are run on a computer with a Intel Core i7 4930K CPU (using one core) and 64G main memory.

A. HSM Combined With BOV Features

In this section, we first study the performance of HSM when BOV features are used.

1) Scene 15: In Fig. 5, we first investigate how the critical parameter γ in Eq. 7 influence the classification. It is shown that the classification accuracy reaches a peak when $\gamma = 0.2$. Thus, we will use $\gamma = 0.2$ for the following experiments.

In Table II, we show the results (training time, testing time and classification accuracy) of different methods on Scene 15 using BOV feature. 1000 codewords are used for this dataset, following [3], [45]. The time to compute the

TABLE II
RESULTS ON SCENE 15: BOV WITH 1000 VISUAL CODEWORDS

Method	Training(s)	Testing(s)	Accuracy(%)
HSM-ga-χ^2	23.66	8.50	85.51±0.19
HSM-ga-HI	22.15	8.54	**85.60±0.21**
HSM-sl-χ^2	49.54	8.66	85.21±0.33
HSM-sl-HI	48.13	8.65	85.22±0.31
HSM$_{ind}$-ga-χ^2	155.00	8.53	85.23±0.32
HSM$_{ind}$-ga-HI	72.40	8.52	85.36±0.33
HSM-linear	13.00	6.59	83.58±0.42
HSM-exact-χ^2	16181.88	31264.56	85.35
SPM-ga-χ^2	10.36	8.47	83.59±0.28
SPM-ga-HI	13.72	8.47	83.62±0.29
SPM-sl-χ^2	17.09	8.66	83.31±0.33
SPM-sl-HI	16.85	8.65	83.34±0.27
SPM-linear	7.92	6.64	82.42±0.35
Spatial saliency [3]			**85.5**±0.6
Spatial Fisher Vector [45]			85.0±0.8

TABLE III
CLASSIFICATION ACCURACY OF HSM (HSM-GA-χ^2) WITH DIFFERENT SPATIAL PYRAMID (SP) SIZE ON SCENE 15

SP level (region #)	Single layer	All layers
Level 0 (1)	81.61±0.30	81.61±0.30
Level 1 (1+5)	84.09±0.26	84.49±0.33
Level 2 (1+5+25)	84.72±0.23	85.51±0.19

overall A and R are 14.71 and 6.15 seconds, respectively. The time to compute independent A^c and R^c are 29.32 and 135.80 seconds, respectively. A and A^c are computed using multiple processor cores.

All HSM accuracy rates are consistently higher than all the SPM accuracy results, which demonstrates the effectiveness of HSM in providing better spatial matching. HSM outperforms SPM in 80% classes of Scene 15. The first block shows the results of HSM and the second block shows the results of SPM. The difference between the training time of HSM-ga-χ^2 (HI) and SPM-ga-χ^2 (HI) is that HSM needs to compute $\boldsymbol{b}$ from $\boldsymbol{a}$ (Eq. 19), which is similar for "sl" method (needs to do data transformation, Eq. 15). HSM-ga is faster than HSM-sl in the training phase, which is related to the observation that HSM-ga converges faster than HSM-sl. The classification accuracy of HSM using different spatial pyramid layers is shown in Table III.

HSM learned with an overall relationship matrix R (HSM-ga-χ^2, HSM-ga-HI) has slightly better results than HSM learned with independent R^c for each class (HSM$_{ind}$-ga-χ^2, HSM$_{ind}$-ga-HI). This is because the range of values for different R^c vary greatly using the same γ in Eq. 7, which lead to a sub-optimal classification result.

A general purpose SVM solver is not suitable for the proposed HSM kernel. We run one round using a general purpose SVM solver (LIBSVM) to solve HSM with χ^2 kernel (HSM-exact-χ^2). LIBSVM uses precomputed kernel matrices for training and testing, which costs much more time than all the fast learning methods.

Compared with related methods [3], [45], HSM has slightly higher classification accuracy than them. It is worth noting that HSM is more stable than the methods in [3] and [45]. The standard deviation of different HSM algorithms' accuracy rates ranges from 0.2 to 0.4, while [3], [45] have 0.6 and 0.8 standard deviation, respectively.

Finally, we show in Fig. 6 the relationship between regions according to the learned overall matrix R. A region usually has a high correlation with its surrounding regions. This relationship matrix is learned from the Scene 15 dataset where an object can appear (almost) anywhere in an image. When learning from object images, e.g., frontal human faces in Fig. 2, stronger relation can be found between local regions. However, even for scene recognition, the simple neighborhood relationship can consistently improve recognition accuracy over SPM, as shown by our experiments.

In Fig. 7, we show the spatial relationship (the 4×4 split in SP) in different classes using the independent relationship matrix, which has different pattern with each other. For example, in the "kitchen" class, walls appear in the upper part and furniture appear in the lower part, so the top regions have a low correlation with the bottom regions. In the "MITtallbuilding" class, buildings often appear in the lower part of an image, so the lower regions have strong correlations. For the "store" class, because all images are full of similar food items, all regions have some relatively strong correlations.

2) Indoor 67: In Table IV, we show the results on Indoor 67, with 4000 codewords. The time to compute the overall A and R are 604.20 and 6.13 seconds, respectively. The time to compute independent A^c and R^c are 1392.51 and 452.3 seconds, respectively. We see that the time to learn R^c almost increases linearly with the class number, which is not efficient when compared to compute an overall R. Thus, for the following SUN and Pascal VOC dataset, we only test the overall R in HSM.

All HSM methods (the first block) gets better accuracy than all methods using SPM (the second block). HSM outperforms SPM in 74.63% classes of Indoor 67. HSM also shows better results than related methods. The state-of-the-art results using BOV features is [46], which augmented coordinates to SIFT features. We use a codebook size of 4000, which is similar to but smaller than theirs (4096). The proposed HSM-linear approach (50.25%) achieves comparable accuracy to theirs (50.22%) using the linear classifier, and nonlinear HSM results are 4% higher (54.44%).

3) SUN: SUN is a large vision dataset involving about 20,000 images in both the training and the testing processes. 4000 visual codewords are used. The time to compute the overall A and R are 5242.08 and 6.10 seconds, respectively.

The results are reported in Table V. HSM (results in the first block) always achieves better accuracy than SPM (results in the second block). HSM outperforms SPM in 76.07% classes of SUN. In [34], different features are used to evaluate on this dataset. The accuracy using dense SIFT feature is 21.5%, which only use 300 visual codewords (fewer than our experiments) to compute BOV. When multiple features are used altogether, the accuracy reaches 38.0%. We also expect HSM to achieve higher accuracy using the more powerful features.

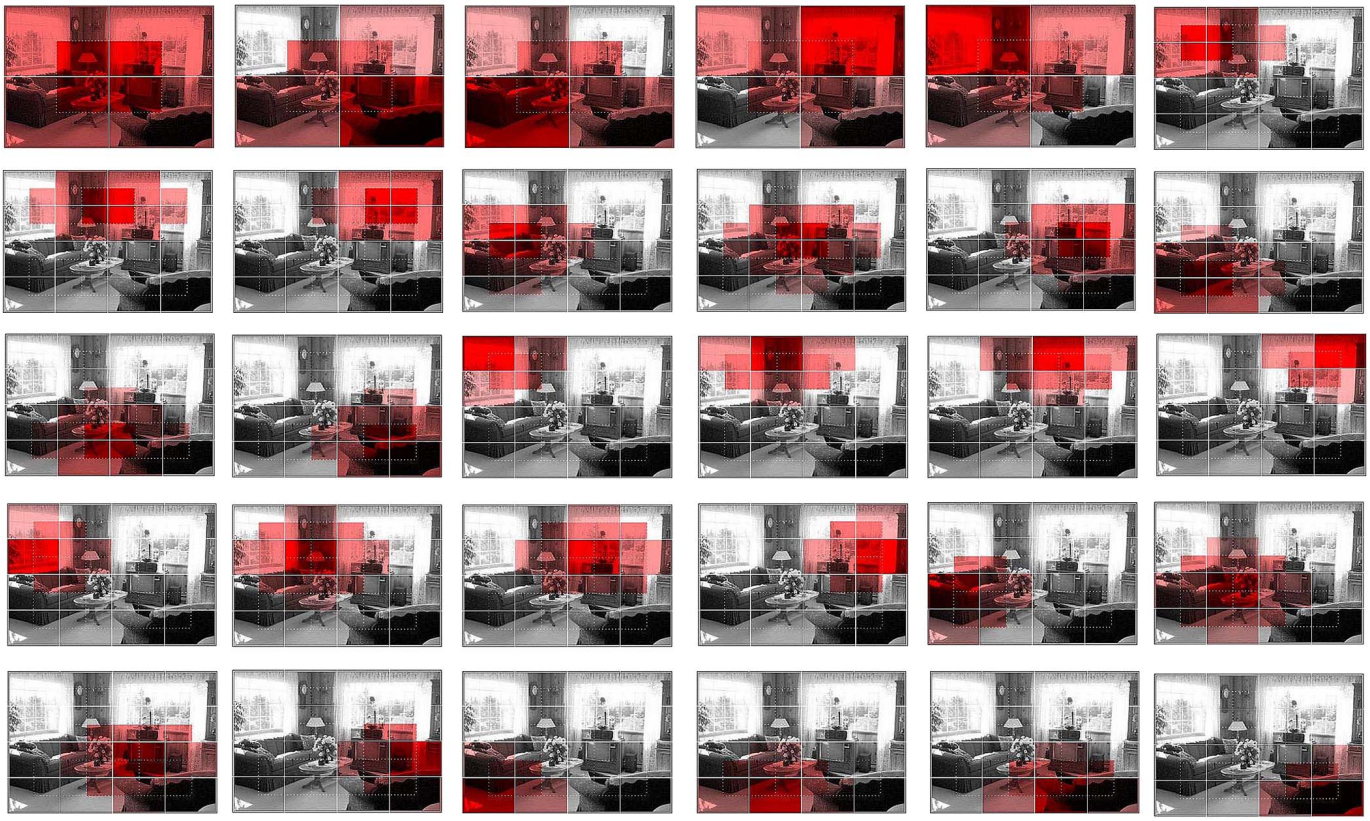


Fig. 6. The relationship of a region (shown in dark red) to other spatial regions. The regions with light red color have close correlation with the dark red region according to the relationship matrix R learned from all training images. This figure should be viewed in color.

TABLE IV
RESULTS ON INDOOR 67: BOV WITH 4000 VISUAL CODEWORDS

Method	Training(s)	Testing(s)	Accuracy(%)
HSM-ga-χ^2	896.02	14.77	53.98±0.68
HSM-ga-HI	768.99	14.63	**54.44±0.71**
HSM-sl-χ^2	1492.91	22.16	53.14±0.65
HSM-sl-HI	1471.29	22.07	53.47±0.69
HSM$_{ind}$-ga-χ^2	2045.331	15.33	53.52±0.69
HSM$_{ind}$-ga-HI	2432.30	14.93	53.93±0.62
HSM-linear	295.63	10.71	50.25±0.69
HSM-exact-χ^2	599472.03	144550.15	53.43
SPM-ga-χ^2	300.29	14.63	52.56±0.59
SPM-ga-HI	328.75	14.64	52.89±0.67
SPM-sl-χ^2	460.59	21.94	52.37±0.86
SPM-sl-HI	460.59	22.01	52.56±0.84
SPM-linear	270.22	10.96	48.58±0.54
BOVW+aug. [46]			50.22

TABLE V
RESULTS OF SUN: BOV WITH 4000 VISUAL CODEWORDS

Method	Training(s)	Testing(s)	Accuracy(%)
HSM-ga-χ^2	3711.59	382.96	**35.89±0.15**
HSM-ga-HI	3691.44	383.41	35.88±0.18
HSM-linear	2419.63	248.30	32.10±0.23
SPM-ga-χ^2	2539.59	386.33	33.98±0.13
SPM-ga-HI	2717.42	384.28	34.18±0.19
SPM-linear	2128.53	266.34	30.53±0.22

B. Hyper-Spatial Features

After empirically validating the performance of HSM, we then evaluate the proposed hyper-spatial features (HSF) on top of BOV (31 regions) and two high-dimensional features: Fisher vector (FV) [32] and VLAD [33].

We use the SPM structure with 8 spatial regions: 1×1, 2×2, and 3×1 for FV and VLAD. SIFT is used as the base feature, which is reduced to 64 dimensions by PCA. We extract FV with 100 Gaussian mixture models (GMM) and VLAD with 100 visual codewords. Only the mean and standard deviation parts in FV are used, following [32]. Thus, one image has $8\times2\times100\times64 = 102400$ dimensions in the FV representation $8\times100\times64 = 51200$ dimensions in VLAD.

One virtue of HSF is that it is almost "free". For example, the appended HSF features have only $\frac{8\times7}{2} = 28$ values for FV and VLAD. That is, appending HSF to FV or VLAD only increases the dimensionality by 0.027% and 0.055%, respectively. These small increases have negligible effect on the storage size, the training and testing time. HSF experimental results are shown in Table VI.

It is obvious that HSF consistently improves SPM, albeit at a smaller improvement scale when compared with that of HSM vs. SPM. However, we want to emphasize that the HSF improvement is noticeable for two reasons. First, HSF improves upon BOV / FV / VLAD almost "for free" (i.e., with negligible additional cost). Second, the improvements are

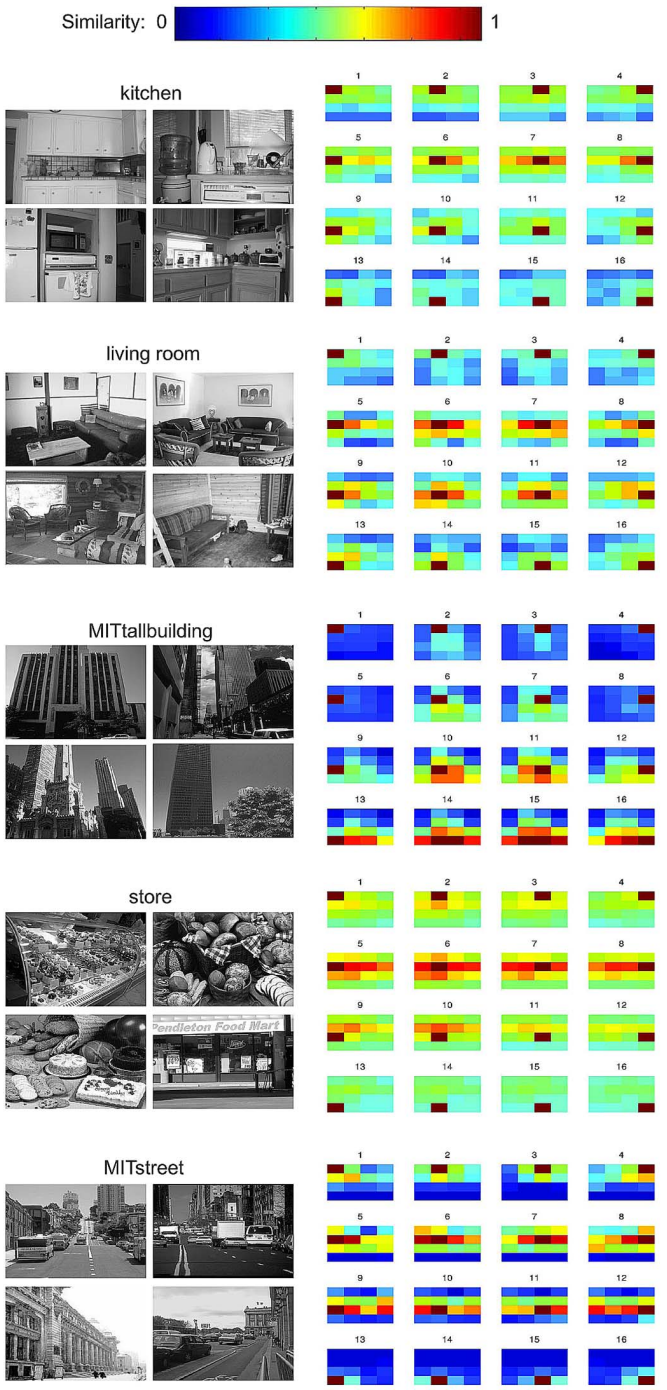


Fig. 7. Relationship of spatial regions in different classes of the Scene 15 dataset, visualized from the learned independent relationship matrix. Each image is divided into 4 by 4 regions. For each region (dark red), the similarities of other regions to it are shown in different color. This figure should be viewed in color.

consistent and significant: BOV (FV/VLAD) + SPM + HSF outperforms BOV (FV/VLAD) + SPM in all the $5\times3\times3=45$ runs in Table VI; and the differences on all three datasets are significant by paired t-test at significance level 0.05.

In the FV experiment, we use the same number of GMM as the setup of the best result in [45]. They achieved the best accuracy $88.2 \pm 0.6\%$ on the Scene 15 dataset. Our result $88.21 \pm 0.21\%$ on this dataset is similar to theirs. We also implement the AUG method [40] and test on three datasets in Table VII, whose results are inferior to HSF, when they use FV with the same number of GMM.

TABLE VI
HSF CLASSIFICATION ACCURACY (%) ON BOV, FV AND VLAD

Datasets	BOV+SPM	BOV+SPM+HSF	t-test
Scene 15	82.42±0.35	82.71±0.37	1.1×10^{-3}
Indoor 67	48.58±0.54	48.81±0.44	3.4×10^{-2}
SUN	30.53±0.22	30.91±0.34	7.9×10^{-3}
Datasets	FV+SPM	FV+SPM+HSF	t-test
Scene 15	87.86±0.29	88.21±0.21	7.6×10^{-3}
Indoor 67	58.85±0.64	59.35±0.53	2.0×10^{-3}
SUN	38.57±0.27	38.82±0.29	1.6×10^{-3}
Datasets	VLAD+SPM	VLAD+SPM+HSF	t-test
Scene 15	87.58±0.25	87.93±0.17	3.8×10^{-2}
Indoor 67	54.38±0.44	55.16±0.70	2.7×10^{-2}
SUN	34.77±0.43	35.46±0.52	1.1×10^{-4}

TABLE VII
ACCURACY OF FV USING AUG METHOD [40]

Datasets	Scene 15	Indoor 67	SUN
Accuracy (%)	87.23	54.11	37.05

TABLE VIII
RESULTS ON VOC2007: BOV WITH 2000 VISUAL CODEWORDS

Method	Training(s)	Testing(s)	mAP(%)
HSM-ga-χ^2	271.89	37.66	54.40
HSM-ga-HI	251.76	37.70	**54.69**
HSM-linear	137.52	34.45	49.63
SPM-ga-χ^2	137.40	37.65	53.15
SPM-ga-HI	170.68	31.71	53.39
SPM-linear	100.63	34.60	48.56
Spatial Fisher Vector (BOV) [45]			52.9

TABLE IX
mAP ON VOC2007: BOV WITH 2000 VISUAL CODEWORDS AND 8-BINS SPM $(1+2\times2+3\times1)$

Method	HSM	SPM
ga-χ^2	54.84	54.58
ga-HI	54.87	54.69
linear	46.24	46.15

C. Multi-Label Image Classification

We evaluate both HSM and HSF on the VOC 2007 dataset, which is a multi-label object categorization problem. We first test HSM + BOV using 2000 visual codewords (same as the setup in [45]). The results of HSM (learned using the gradient approximation strategy) and SPM are presented in Table VIII. HSM's results outperform that of SPM, with the same base kernel and the same BOV features. The time to compute the overall A and R are 341.01 and 6.04 seconds, respectively. In Table IX, we test a different SPM structure with BOV using 8 regions: $1+2\times2+3\times1$. HSM also leads better accuracy than SPM, although the improvement is less than that in Table VIII using 31 spatial regions.

TABLE X
RESULTS ON VOC2007 USING FV WITH 256 GMMS

Method	mAP (%)
FV (GMM 256)+SPM+HSF	**61.65**
FV (GMM 256)+SPM	61.42
Spatial Fisher Vector (GMM 200) [45]	56.5
Spatial Fisher Vector (GMM 500) [45]	56.6
AUG (GMM 256) [40]	**62.0**

We also evaluate HSF + FV on the VOC 2007 dataset. The results are presented in Table X. We use 256 GMMs (following the setup of the AUG method [40]). The HSF result is close to that of AUG, which augmented the coordinates to local features and then computed FV.

D. Discussions

After presenting experimental results and related observations, we now provide some more subjective comments to summarize the findings of this paper, and to discuss the limitations and drawbacks of the proposed methods.

First, the focus of hyper-spatial matching is to take into account of matching between regions at different spatial locations. Concerning spatial matching in image similarity computation, we find that:

- Although Spatial Pyramid Matching (SPM) and almost all of its improvements focused on matching between regions at the same spatial locations in two images, regions at different spatial locations have significant correlations that cannot be ignored (cf. Fig. 3);
- In image classification, the proposed hyper-spatial matching (HSM) consistently achieves higher accuracy than SPM by explicitly considering relationships among all spatial regions (cf. Tables II–V);
- HSF, a simplified version of HSM, considers the relationship between spatial regions in one image. HSF consistently improves the classification ability for very long features (cf. Table VI and Table X), but almost does not incur additional cost in memory or CPU usage.
- In the relationship learned by HSM from scene images, a region only has non-trivial relationship with spatially nearby regions, and the relationship strength at the same spatial location is the strongest amongst all regions. In this sense, SPM intuitively captures the most important components of all spatial relationships in HSM (cf. Fig. 6 and Fig. 7);
- Different image categories exhibits varied spatial relationships (cf. Fig. 7), which might be used in the future for other tasks, e.g., object recognition; and,
- The learned spatial relationships are stable inside a specific problem or dataset (cf. Section II-B) and can be efficiently computed even for large scale datasets.

Furthermore, we have proposed two learning strategies (gradient approximation "ga" and spectral linearization "sl") for HSM SVM. We also presented image classification results with either an overall relation R or multiple relationship matrices R^c. The following findings can be concluded concerning HSM SVM learning:

- The effectiveness of relationship matrices (R or R^c) can be explained by the fact that when it is used in the HSM kernel, it effectively decorrelates the relationships among regions at different spatial locations (cf. Fig. 3);
- The HSM-ga strategy (cf. Algorithm 3) consistently achieves the best accuracy and efficiency during both training and testing (cf. Tables II-V). In image classification with the hyper-spatial matching kernel, HSM-ga is the preferred method;
- HSM-sl (cf. Algorithm 1), on the other hand, is inferior in terms of practical performance. However, this strategy provides convenient tools to understand and visualize spatial matching (cf. Fig. 3). And, while HSM-ga is mostly confined to be used with SVM learning, explicit mapping (cf. Eq. 15) has applications in wider domains;
- One more note is that although we present the HSM framework and learning strategies based on SPM, it has wider range of applications. For example, it can also be combined with the receptive fields in [4].

In our experience, there is one major limitation of the proposed learning strategies:

- **Memory Consumption for Large Scale Datasets**. As described at the beginning of this section, we store some precomputed values in memory, which is an $n \times Nd$ dense array. The input data is also $n \times Nd$, but could well be sparse. It is thus important to research on how to remove this memory assumption dependency in HSM-ga. On the other hand, HSM-sl requires $n \times (2m+1)Nd$ memory to store the linearized dataset, which has even higher storage requirement.

V. CONCLUSIONS AND FUTURE WORK

In this paper, we propose hyper-spatial matching (HSM), a framework for flexibly matching two images. HSM takes into account the interactions among all pairs of spatial regions in two images, which alleviates the mismatching problem in SPM. Thus, HSM provides more flexibility and better similarity measures for comparing two images than SPM.

We use relationship matrices to capture the relationship or correlation between any two spatial locations of all images. Training images are used to learn two types of relationship matrices. The first is one overall relationship matrix R for a dataset, and the second type contains multiple matrices, one relationship matrix R^c per image class. To deal with the high computational cost induced by HSM, we propose two fast learning strategies for HSM SVM learning and testing. HSM-ga is based on the dual coordinate descent SVM framework, where polynomial regression is used to approximate the SVM gradient on each dimension of the feature vector. It is hundreds of times faster than a general purpose SVM solver in training and testing without loss of accuracy. In the experiment, we compare HSM-ga with state-of-the-art methods on three large scale scene datasets and one mutli-label image classification problem. The proposed fast classifier with HSM kernel shows better classification accuracy than compared methods.

The other proposed learning strategy is HSM-sl, which provides an approximate linearization strategy for HSM and uses the feature mapping approach for additive kernels. Although HSM-sl is inferior to HSM-ga in practical performance, it provides ways to study the spatial matching behaviors. HSM-sl reveals that the hyper-spatial matching kernel is effective because it decorrelates the interactions among spatial regions at different locations.

For very long features like Fisher vector and VLAD, HSF is proposed to approximate HSM. HSF computes the relationship between all spatial regions in one image. By appending HSF to the original features, the classification accuracy is improved consistently without additional cost in memory and CPU usage.

In the future, our works include the following issues. First, the HSM-sl linearization can provide better representation for an image, which might have further applications in applications such as image retrieval. Second, when we study problems with more meaningful spatial relationships, the proposed relationship matrix and learning strategies could be of more use, e.g., in object detection where the object of interest has strong spatial arrangement in it. Finally, we will consider to learn the relationship matrix and the classifier together to get a more uniform framework.

ACKNOWLEDGMENT

The authors would like to thank the anonymous reviewers for their useful comments and suggestions.

REFERENCES

[1] S. Lazebnik, C. Schmid, and J. Ponce, "Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, vol. 2. 2006, pp. 2169–2178.

[2] T. Harada, Y. Ushiku, Y. Yamashita, and Y. Kuniyoshi, "Discriminative spatial pyramid," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2011, pp. 1617–1624.

[3] G. Sharma, F. Jurie, and C. Schmid, "Discriminative spatial saliency for image classification," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2012, pp. 3434–3441.

[4] Y. Jia, C. Huang, and T. Darrell, "Beyond spatial pyramids: Receptive field learning for pooled image features," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2012, pp. 3370–3377.

[5] Y.-L. Boureau, F. Bach, Y. LeCun, and J. Ponce, "Learning mid-level features for recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2010, pp. 2559–2566.

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

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

[8] F. Perronnin, J. Sánchez, and T. Mensink, "Improving the Fisher kernel for large-scale image classification," in *Proc. 11th Eur. Conf. Comput. Vis.*, Sep. 2010, pp. 143–156.

[9] J. Wu and J. M. Rehg, "CENTRIST: A visual descriptor for scene categorization," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 8, pp. 1489–1501, Aug. 2011.

[10] F. Perronnin and C. Dance, "Fisher kernels on visual vocabularies for image categorization," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2007, pp. 1–8.

[11] 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, Aug. 2008.

[12] J. Wu, "Efficient HIK SVM learning for image classification," *IEEE Trans. Image Process.*, vol. 21, no. 10, pp. 4442–4453, Oct. 2012.

[13] A. Vedaldi and A. Zisserman, "Efficient additive kernels via explicit feature maps," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 3, pp. 480–492, Mar. 2012.

[14] J. Wu, "Power mean SVM for large scale visual classification," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2012, pp. 2344–2351.

[15] G. Sharma and F. Jurie, "Learning discriminative spatial representation for image classification," in *Proc. Brit. Mach. Vis. Conf.*, 2011.

[16] Y. Jiang, J. Yuan, and G. Yu, "Randomized spatial partition for scene recognition," in *Proc. 12th Eur. Conf. Comput. Vis.*, Oct. 2012, pp. 730–743.

[17] H. Bilen, V. Namboodiri, and L. Van Gool, "Object and action classification with latent variables," in *Proc. Brit. Mach. Vis. Conf.*, Sep. 2011.

[18] D. Xu, S. Yan, and J. Luo, "Face recognition using spatially constrained earth mover's distance," *IEEE Trans. Image Process.*, vol. 17, no. 11, pp. 2256–2260, Nov. 2008.

[19] J. Zou, Q. Ji, and G. Nagy, "A comparative study of local matching approach for face recognition," *IEEE Trans. Image Process.*, vol. 16, no. 10, pp. 2617–2628, Oct. 2007.

[20] S. Gao, I. W.-H. Tsang, and L.-T. Chia, "Laplacian sparse coding, hypergraph Laplacian sparse coding, and applications," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 1, pp. 92–104, Jan. 2013.

[21] A. Vedaldi and S. Soatto, "Relaxed matching kernels for robust image comparison," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2008, pp. 73–86.

[22] E. S. Ng and N. G. Kingsbury, "Robust pairwise matching of interest points with complex wavelets," *IEEE Trans. Image Process.*, vol. 21, no. 8, pp. 3429–3442, Aug. 2012.

[23] A. Perina and N. Jojic, "Spring lattice counting grids: Scene recognition using deformable positional constraints," in *Proc. Eur. Conf. Comput. Vis.*, 2012, pp. 837–851.

[24] O. Russakovsky, Y. Lin, K. Yu, and L. Fei-Fei, "Object-centric spatial pooling for image classification," in *Proc. 12th Eur. Conf. Comput. Vis.*, Oct. 2012, pp. 1–15.

[25] A. Perina and N. Jojic, "Image analysis by counting on a grid," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2011, pp. 1985–1992.

[26] X. Zhang, Z. Li, L. Zhang, W.-Y. Ma, and H.-Y. Shum, "Efficient indexing for large scale visual search," in *Proc. IEEE 12th Int. Conf. Comput. Vis.*, Sep./Oct. 2009, pp. 1103–1110.

[27] T. de Campos, G. Csurka, and F. Perronnin, "Images as sets of locally weighted features," *Comput. Vis. Image Understand.*, vol. 116, no. 1, pp. 68–85, Jan. 2012.

[28] S. Maji, A. C. Berg, and J. Malik, "Efficient classification for additive kernel SVMs," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 1, pp. 66–77, Jan. 2013.

[29] F. Perronnin, Z. Akata, Z. Harchaoui, and C. Schmid, "Towards good practice in large-scale learning for image classification," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2012, pp. 3482–3489.

[30] G. Sharma and F. Jurie, "A novel approach for efficient SVM classification with histogram intersection kernel," in *Proc. Brit. Mach. Vis. Conf.*, 2013.

[31] F. Perronnin, J. Sánchez, and Y. Liu, "Large-scale image categorization with explicit data embedding," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2010, pp. 2297–2304.

[32] F. Perronnin and C. Dance, "Fisher kernels on visual vocabularies for image categorization," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2007, pp. 1–8.

[33] H. Jegou, F. Perronnin, M. Douze, J. Sanchez, P. Perez, and C. Schmid, "Aggregating local image descriptors into compact codes," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 9, pp. 1704–1716, Sep. 2012.

[34] J. Xiao, J. Hays, K. A. Ehinger, A. Oliva, and A. Torralba, "SUN database: Large-scale scene recognition from abbey to zoo," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2010, pp. 3485–3492.

[35] D. Haussler, "Convolution kernels on discrete structures," Dept. Comput. Sci., Univ. California Santa Cruz, Santa Cruz, CA, USA, Tech. Rep. UCSC-CRL-99-10, 1999.

[36] C. Wallraven and B. Caputo, "Recognition with local features: The kernel recipe," in *Proc. IEEE Int. Conf. Comput. Vis.*, Oct. 2003, pp. 257–264.

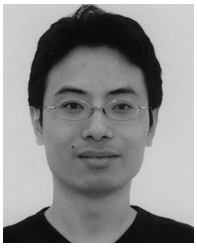
[37] S. Lyu, "Mercer kernels for object recognition with local features," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, Jun. 2005, pp. 223–229.

[38] L. Bo and C. Sminchisescu, "Efficient match kernels between sets of features for visual recognition," in *Proc. Adv. Neural Inf. Process. Syst.*, 2009, pp. 135–143.

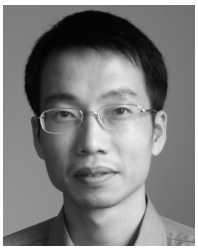
[39] G.-X. Yuan, C.-H. Ho, and C.-J. Lin, "Recent advances of large-scale linear classification," *Proc. IEEE*, vol. 100, no. 9, pp. 2584–2603, Sep. 2012.

[40] J. Sánchez, F. Perronnin, and T. de Campos, "Modeling the spatial layout of images beyond spatial pyramids," *Pattern Recognit. Lett.*, vol. 33, no. 16, pp. 2216–2223, Dec. 2012.
[41] M. C. Grant and S. Boyd. *CVX: Matlab Software for Disciplined Convex Programming*. [Online]. Available: http://cvxr.com/cvx, accessed Jul. 12, 2013.
[42] C.-C. Chang and C.-J. Lin, "LIBSVM: A library for support vector machines," *ACM Trans. Intell. Syst. Technol.*, vol. 2, no. 3, pp. 27:1–27:27, 2011. [Online]. Available: http://www.csie.ntu.edu.tw/~cjlin/libsvm
[43] A. Quattoni and A. Torralba, "Recognizing indoor scenes," in *Proc. Comput. Vis. Pattern Recognit.*, 2009, pp. 413–420.
[44] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. (2007). *The PASCAL Visual Object Classes Challenge Results*. [Online]. Available: http://www.pascal-network.org/challenges/VOC/voc2007/workshop/index.html
[45] J. Krapac, J. Verbeek, and F. Jurie, "Modeling spatial layout with fisher vectors for image categorization," in *Proc. Int. Conf. Comput. Vis. Pattern Recognit.*, 2012, pp. 1487–1494.
[46] A. Vedaldi and B. Fulkerson. (2008). *VLFeat: An Open and Portable Library of Computer Vision Algorithms*. [Online]. Available: http://www.vlfeat.org/

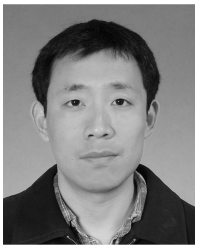
Yu Zhang received the B.S. and M.S. degrees in telecommunications engineering from Xidian University, Xi'an, China. He is currently pursuing the Ph.D. degree with the School of Computer Engineering, Nanyang Technological University, Singapore. His research interest is computer vision.



Jianxin Wu (M'09) received the B.S. and M.S. degrees in computer science from Nanjing University, Nanjing, China, and the Ph.D. degree in computer science from the Georgia Institute of Technology, Atlanta, GA, USA. He is currently a Professor with the Department of Computer Science and Technology, Nanjing University, and is with the National Key Laboratory for Novel Software Technology, Nanjing University. He was an Assistant Professor with Nanyang Technological University, Singapore. His research interests are computer vision and machine learning.



Jianfei Cai (S'98–M'02–SM'07) received the Ph.D. degree from the University of Missouri-Columbia, Columbia, MO, USA. He is currently an Associate Professor, and has served as the Head of the Visual and Interactive Computing Division and the Computer Communication Division at the School of Computer Engineering, Nanyang Technological University, Singapore. His major research interests include visual computing and multimedia networking. He has authored more than 140 technical papers in the international journals and conferences. He has been actively involved in program committees of various conferences. He has served as the leading Technical Program Chair of the 2012 IEEE International Conference on Multimedia and Expo and the leading General Chair of the 2012 Pacificrim Conference on Multimedia. He was an invited speaker of the first IEEE Signal Processing Society Summer School on 3-D and high-definition/high-contrast video process systems in 2011. Since 2013, he has served as an Associate Editor of the IEEE TRANSACTIONS ON IMAGE PROCESSING. He also served as an Associate Editor of the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY from 2006 to 2013, and a Guest Editor of the IEEE TRANSACTIONS ON MULTIMEDIA and Elsevier's *Journal of Visual Communication and Image Representation*.



Weiyao Lin received the B.E. and M.E. degrees from Shanghai Jiao Tong University, Shanghai, China, in 2003 and 2005, respectively, and the Ph.D. degree from the University of Washington, Seattle, WA, USA, in 2010, all in electrical engineering. He is currently an Associate Professor with the Department of Electronic Engineering, Shanghai Jiao Tong University.

His research interests include video processing, machine learning, computer vision, and video coding and compression.