Random Decomposition Forests

Chun-Han Chien and Hwann-Tzong Chen
Department of Computer Science
National Tsing Hua University, Taiwan

Abstract

We present an effective image representation based on a new tree-structured coding technique called 'random decomposition forests' (RDFs). Our method combines the merits of visual-word representations and random forests. The proposed RDF is able to decompose a local descriptor into multiple sets of visual words in a recursive and randomized manner. We show that, when combined with standard multiscale and spatial pooling strategies, the code vectors generated by RDF yield a powerful representation for image categorization. We are able to achieve state-of-the-art performance on several popular benchmark datasets.

1. Introduction

Image categorization is one of the key problems in computer vision and pattern recognition. Typically, the first step toward solving the problem of image categorization is to design an effective image representation. The *bag-of-words* model is a well-known image representation. The model represents an image as a histogram of visual words. The visual words are obtained by performing clustering or vector quantization on low-level feature descriptors such as SIFT [8] or HOG [2]. From an image we may thus extract a lot of features and then quantize those features into visual words to build a histogram that represents the image.

A drawback of the bag-of-words model is that it is prone to lose the underlying local descriptor's power due to the process of quantization. Another limitation is the disposal of the context and the spatial information. Nonlinear kernels or hierarchical clustering may be used to alleviate the loss of descriptor's power. On the other hand, to incorporate spatial information, the *spatial pyramid matching* (SPM) approach [6] may be adopted. SPM partitions an image into a grid of $2^l \times 2^l$ spatial bins ($l = 0, 1, 2$), and then computes the bag-of-words within each of the spatial bin. All of the histograms are concatenated together as a final representation. It has been shown that the combination of bag-of-words and spatial pyramid matching can achieve good performance on many classification tasks.

Later methods such as *sparse coding spatial pyramid matching* (ScSPM) [15] and *locality-constrained linear coding* (LLC) [14] seek to come up with better coding schemes and local pooling strategies. The encoding step is to find the histogram of local features. The pooling step is to aggregate the histograms. In these respects, the ScSPM [15] method uses sparse coding to represent local features, and the LLC [14] method further improves ScSPM by imposing locality constraints on sparse coding. Unlike the bag-of-words model, both of ScSPM and LLC are effective enough so that they can achieve good performance using just simple classifiers, *e.g.* SVM with a linear kernel, to train and predict with the sparse representations. The use of linear classifiers also greatly reduces computation time for training the model and for performing prediction on test data.

We present a new image representation called *random decomposition forest* (RDF) model which uses multiple randomized trees to encode local feature descriptors. On each randomized tree in the forest, an input feature descriptor would start from the root and then follow a path to reach some leaf node. The path corresponds to a specific way of decomposing the input descriptor. The proposed RDF model provides a seamless integration of unsupervised and supervised feature learning: At each node, we compute the projection of the input descriptor onto a visual word, and propagate the reconstruction residue to a subtree, in a manner similar to matching pursuit (unsupervised learning). The splitting criterion for growing subtrees is similar to the standard random forest algorithm, that is, we seek to separate the current training data so that the class impurity could be reduced after splitting (supervised learning). The encoding outputs of different trees are concatenated into a single representation. An advantage of using random forests is that the trees are independent. The descriptor can go through all trees simultaneously, which is suitable for parallel processing and easy to scale up. Furthermore, each tree implies a specific dictionary subset, and every descriptor would be independently reconstructed from different dictionary subsets. Since the descriptor is represented in multiple ways, we may get more information about the descriptor.

1.1. Related Work

Boiman *et al*. [1] present a nearest-neighbor-based classifier called *naive Bayes nearest-neighbor* (NBNN). They propose to keep all descriptors in the original feature space instead of quantizing the descriptors into visual words since the quantization done by the bag-of-words model might decrease the discrimination power of descriptors. NBNN has several variants, e.g. the NBNN kernel [13] and local NBNN [10]. Tuytelaars *et al*. [13] propose a kernelized version of NBNN. The NBNN kernel can be used with other kernels in multiple kernel learning (MKL) to increase the performance. McCann and Lowe [10] propose the local NBNN which improves NBNN by searching only the local neighbors of the descriptor instead of finding neighbors in each class. The NBNN kernel and local NBNN have better performance than the original NBNN. In the experiments we also compare our method with SVM-KNN [16], which combines nearest-neighbor classification with SVM and performs quite well on image categorization tasks.

The bag-of-words model can be combined with *spatial pyramid matching* (SPM) [6]: First, local feature descriptors, such as SIFT or HOG, are densely extracted from the input image. Then a dictionary with M entries (or visual words) is used to quantize each descriptor. The descriptors are converted into a code vector $\mathbf{v} \in \mathbb{R}^M$. Quantization can be done by hard assignment or soft assignment. In hard assignment, each descriptor is assigned to one dictionary entry, and the code vector has only one nonzero component. On the other hand, soft assignment allows one descriptor to be assigned to a group of visual words, so the code vector has more than one non-zero component. Second, the image is partitioned into sub-regions, and the code vectors inside each sub-region are pooled together to build a normalized histogram. Then, the image representation is obtained by concatenating the histograms from all sub-regions. However, due to quantization, the discriminative power of descriptors decreases and it often requires nonlinear classification algorithms to train and predict with the image representations. The need of nonlinear classification might imply poorer scalability. *Sparse coding spatial pyramid matching* (ScSPM) [15] and *locality-constrained linear coding* (LLC) [14] are two methods to improve the scalability as well as the accuracy. ScSPM replaces quantization by sparse coding and enforces using only a small number of visual words to reconstruct a descriptor. LLC exploits the property that a descriptor is more similar to its neighboring visual words than farther visual words, and therefore it may be more suitable to encode a descriptor by its neighboring visual words.

Regarding tree-based feature learning, Moosmann *et al*. [11] propose to learn visual codebooks using *extremely randomized clustering forests*. Our method is different from the method of Moosmann *et al*. in that we integrate visual-word decomposition with the construction of random forests. Our work also shares the same spirit of tree-structured visual coding as [12], although the overall approaches are quite different: the vocabulary tree is built by hierarchical k-means clustering, and thus is an unsupervised approach.

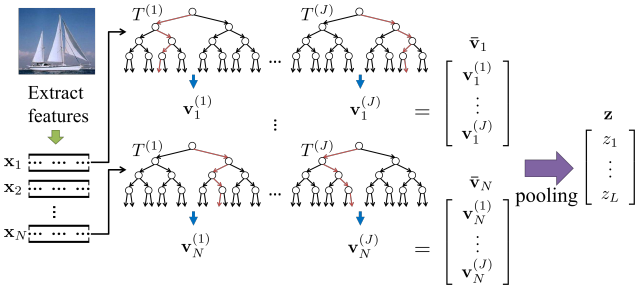

Figure 1. A random decomposition forest (RDF) transforms an image into a representation $\mathbf{z} = [z_1 \dots z_L]^\mathsf{T}$. First, local feature descriptors $\mathbf{x}_1, \dots, \mathbf{x}_N$ are densely extracted from the image. Second, each descriptor $\mathbf{x}_i$ is sent to random trees, $T^{(1)}, \dots, T^{(J)}$, and each tree $T^{(j)}$ would produce a code vector $\mathbf{v}_i^{(j)}$ for $\mathbf{x}_i$. We concatenate the vectors $\mathbf{v}_i^{(1)}, \dots, \mathbf{v}_i^{(J)}$ to form a big code vector $\bar{\mathbf{v}}_i = \left[\mathbf{v}_i^{(1)\mathsf{T}} \dots \mathbf{v}_i^{(J)\mathsf{T}}\right]^\mathsf{T}$. Finally $\bar{\mathbf{v}}_1, \dots, \bar{\mathbf{v}}_N$ are aggregated via spatial pooling to produce an image representation $\mathbf{z}$.

2. Random Decomposition Forests

Our method contains two main stages: *i*) dictionary learning and *ii*) random-forest encoding. In the first stage, we try to find candidate visual words that are good for representing local descriptors. We use sparse coding to learn the dictionary, and the learned dictionary would be used in the second stage. It is worth noting that our method is not bound to a particular dictionary learning algorithm. Any dictionary learning algorithm can be used in the first stage of our method for generating a large pool of useful visual words. The second stage involves the proposed random decomposition forest (RDF) model. We use a subset of training data to build an RDF. The construction of the RDF will be detailed in Section 2.2. Suppose that we have an already trained RDF. Each of the densely extracted descriptors of an image is converted into a code vector $\bar{\mathbf{v}}$. The dimensionality of code vector $\bar{\mathbf{v}}$ is equal to the total number of all internal nodes of all trees in the RDF. Each dimension of $\bar{\mathbf{v}}$ corresponds to one of the internal nodes, and it has a nonzero value if the descriptor reaches that internal node. After encoding all descriptors in the image, we use spatial max-pooling to derive the final representation. Fig. 1 illustrates how our method transforms an image into a representation. Finally, each training image can be represented by a spatially-pooled code vector $\mathbf{z}$ using the RDF, and we adopt one-versus-all strategy to train linear classifiers over all code vectors. Given a test image, we use the RDF to compute its code vector and predict its class label by the

linear classifiers. Sections 2.1 and 2.2 describe the details of the two stages.

2.1. Dictionary Learning

The idea of dictionary learning is to find a set of visual words that can be used to represent local feature descriptors. Each descriptor could be mapped into M-dimensional code, where M is the number of entries or visual words in the dictionary. Because of the good performance of spare coding, we use sparse coding to learn the dictionary required in our method. Let $\mathbf{X}$ be a set of N local feature descriptors in D-dimensional feature space, extracted from some training images, *i.e.* $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N] \in \mathbb{R}^{D \times N}$. We follow the standard formulation of sparse-coding optimization to compute the dictionary $\mathbf{W}$:

$$\begin{aligned} &\underset{\mathbf{W}, \{\boldsymbol{\alpha}_i\}}{\arg\min} \sum_{i=1}^{N} \|\mathbf{x}_i - \mathbf{W}\boldsymbol{\alpha}_i\|_2^2 + \lambda \|\boldsymbol{\alpha}_i\|_1 \,, \\ &\text{subject to} \, \|\mathbf{w}_j\|_2 \leq 1 \,, \forall j = 1, \ldots, M \,, \end{aligned} \tag{1}$$

where the dictionary $\mathbf{W} = [\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_M] \in \mathbb{R}^{D \times M}$ is a set of M visual words, λ is a trade-off weight, and $\boldsymbol{\alpha}_i \in \mathbb{R}^M$ comprises the coefficients for $\mathbf{x}_i$, for $i = 1, \ldots, N$. In Eq. (1), the dictionary $\mathbf{W}$ combined with coefficient $\boldsymbol{\alpha}_i$ can reconstruct the descriptor $\mathbf{x}_i$, and $\boldsymbol{\alpha}_i$ has to be sparse. We solve Eq. (1) using the SPAMS toolbox [9]. After getting the dictionary, we use the random decomposition forest to choose subsets of visual words, and only employ the chosen visual words to encode the descriptors, as described in the following section.

2.2. RDF Encoding and Construction

Unlike conventional random forests which are used for classification or regression, the random decomposition forest (RDF) is designed to decompose descriptors and to generate discriminative image representations. For ease of explanation, we start with describing the step of RDF descriptor encoding, assuming that the RDF is already available.

RDF descriptor encoding: The step of RDF encoding is to convert the input descriptor into a code vector. A descriptor would traverse each tree of RDF from the root to the leaf, and would be recursively decomposed by the visual words that are associated with the internal nodes. An internal node of a tree in a trained RDF uses two visual words: *i*) the reconstruction visual word $\widehat{\mathbf{w}}$ and *ii*) the splitting visual word $\widetilde{\mathbf{w}}$. For convenience we assume that every visual word has been normalized to be a unit vector. When a (residual) descriptor $\mathbf{x}$ reaches an internal node, we project $\mathbf{x}$ onto the reconstruction visual word $\widehat{\mathbf{w}}$ to get the absolute projected value $\left|\widehat{\mathbf{w}}^\mathsf{T}\mathbf{x}\right|$. More specifically, for internal node m, we compute the code vector component by

$$v_m = \begin{cases} \left|\widehat{\mathbf{w}}^\mathsf{T}\mathbf{x}\right| \,, & \text{if } \mathbf{x} \text{ reaches internal node } m \,, \\ 0 \quad \,, & \text{otherwise} \,. \end{cases} \tag{2}$$

The reconstruction visual word $\widehat{\mathbf{w}}$ is chosen when we build the RDF, which will be explained later. After obtaining v_m, we compute the difference between $\mathbf{x}$ and $(\widehat{\mathbf{w}}^\mathsf{T}\mathbf{x})\widehat{\mathbf{w}}$ to get the residual descriptor: $\mathbf{x}' = \mathbf{x} - (\widehat{\mathbf{w}}^\mathsf{T}\mathbf{x})\widehat{\mathbf{w}}$. We then use another visual word $\widetilde{\mathbf{w}}$ associated with internal node m to decide which child node the residual descriptor $\mathbf{x}'$ should reach next. The split function based on the splitting visual word $\widetilde{\mathbf{w}}$ is expressed as

$$\text{split}\,(\mathbf{x}') : \begin{cases} \text{goes to right child,} & \text{if } \widetilde{\mathbf{w}}^\mathsf{T}\mathbf{x}' \geq \theta \,, \\ \text{goes to left child,} & \text{if } \widetilde{\mathbf{w}}^\mathsf{T}\mathbf{x}' < \theta \,, \end{cases} \tag{3}$$

where the splitting visual word $\widetilde{\mathbf{w}}$ and θ are chosen previously during the construction of RDF. After traversing tree $T^{(j)}$, a descriptor can be encoded as $\mathbf{v}^{(j)} = [v_1, \ldots, v_{|T^{(j)}|}]^\mathsf{T}$, where $|T^{(j)}|$ denotes the number of internal nodes of tree $T^{(j)}$. We run through all J trees in the RDF and concatenate all $\{\mathbf{v}^{(j)}\}_{j=1}^{J}$ to get the code vector $\bar{\mathbf{v}}$. The dimensionality of $\bar{\mathbf{v}}$ is therefore $\sum_{j=1}^{J} |T^{(j)}|$.

Growing an RDF: To build a random decomposition forest, we need to get a large set of local feature descriptors. We extract local feature descriptors from the images of different classes, and then run k-means clustering to partition the descriptors into clusters. From each cluster of each class we chose the descriptor which is closest to the center. Assume that we have C classes, and for each class we have K clusters. We would collect $C \times K$ representative descriptors for building the RDF. The trees are grown in a top-down manner. For each internal node, we need to choose two visual words $\widehat{\mathbf{w}}$ and $\widetilde{\mathbf{w}}$ from a subset of dictionary. We solve the following optimization to get the reconstruction visual word $\widehat{\mathbf{w}}$:

$$\widehat{\mathbf{w}} = \underset{\mathbf{w} \in \mathbf{W}'}{\arg\max} \sum_{\mathbf{x}_i \in Q_m} \left|\mathbf{w}^\mathsf{T}\mathbf{x}_i\right| \,, \tag{4}$$

where $\mathbf{W}' \subset \mathbf{W}$ is a subset of dictionary, which is randomly picked from the original dictionary $\mathbf{W}$. The set Q_m contains the residual descriptors reaching the internal node m. We find the visual word $\widehat{\mathbf{w}}$ that yields the largest total absolute projected values over all residual descriptors in Q_m. The idea is borrowed from matching pursuit.

Another visual word $\widetilde{\mathbf{w}}$ is required by the split function defined in Eq. (3) to split the residual descriptors into two child nodes. Similar to the standard split criterion in random forests, we use information gain to evaluate different split functions. The information gain is computed as follows:

$$\Delta H = H(Q_m) - \sum_{i \in \{1,2\}} \frac{|Q_m^i|}{|Q_m|} H\left(Q_m^i\right) \,, \tag{5}$$

where Q_m is the set of descriptors in node m, and Q_m^i is the subset of descriptors sent to branch i of node m. Here we only consider binary split so $i = 1, 2$. The operator $|\cdot|$ denotes the size of set. We have $H\left(Q_m^i\right)$ as the entropy of set Q_m^i, which can be computed by $H\left(Q_m^i\right) = -\sum_{j=1}^{C} p_j \log\left(p_j\right)$, where p_j is the proportion of descriptors in Q_m^i belonging to the jth class. The total number of classes is C. From a subset of dictionary, we would choose the visual word $\widetilde{\mathbf{w}}$ with an optimal threshold θ to separate the descriptors using Eq. (3) such that the resulting branches Q_m^1 and Q_m^2 maximize Eq. (5). The process of growing a tree is done in a recursive manner. The recursion stops when the number of descriptors in the node is smaller than a predefined threshold, or when the height of tree exceeds a certain value. We repeat the same process to build multiple trees and form a forest. Each tree provides a specific way of decomposing the descriptor via exploiting the reconstruction quality and the class information through the tree structure.

To derive the representation of an image, we extract dense local descriptors from the image. We feed all descriptors $\{\mathbf{x}_i\}$ into the RDF and obtain the code vectors $\{\bar{\mathbf{v}}_i\}$. To aggregate the code vectors, we partition the image into a grid of $2^l \times 2^l$ spatial bins ($l = 0, 1, 2$), and then apply max-pooling within each bin. The max-pooling operation is done by taking the maximum value component-wise over the code vectors $\{\bar{\mathbf{v}}|\bar{\mathbf{v}} \in \mathcal{S}\}$ in the spatial bin $\mathcal{S}$. The result of max-pooing for each spatial bin is thus a vector of the same dimensionality as $\bar{\mathbf{v}}$, which is $\sum_{j=1}^{J} |T^{(j)}|$, providing that there are J trees in the RDF and $|T^{(j)}|$ is the number of internal nodes of tree $T^{(j)}$. Finally, all of the pooling results are concatenated together to produce an image representation $\mathbf{z}$. The dimensionality of $\mathbf{z}$ is $21 \times \sum_{j=1}^{J} |T^{(j)}|$, since we have $1 + 4 + 16 = 21$ spatial bins. We use linear SVM to train multi-class classifiers on the derived image representations. We adopt the one-against-all setting for multiclass classification. In our implementation, we use the LIBLINEAR package [3] to train linear SVMs.

3. Experimental Results

We use only SIFT [8] as the local-descriptor extractor throughout the experiments, although any local descriptor can be used with our RDF model. Inspired by [14], we extract SIFT descriptors at multiple scales, namely, patches of 16×16, 24×24, and 32×32 pixels. The patches of different scales are densely sampled on a grid with step-size of 6 pixels. We evaluate the RDF representation using three popular benchmark datasets for image categorization: the Fifteen Scenes dataset [7], Caltech-101 [4], and Caltech-256 [5], summarized as follows.

Fifteen Scenes: The Fifteen Scenes dataset [7] contains 4,485 images in 15 different kinds of scenes, and each kind of scene has a number of images from 200 to 400. The typical size of the image is 300×250 pixels. The dataset includes both outdoor and indoor images, such as bedroom, street, *etc*. According to the setting of [6] , we take 100 images per class to train our model, and the rest to evaluate the performance.

Caltech-101: The Caltech-101 dataset [4] consists of 9,144 images in 102 categories, *i.e*. 101 object classes and a 'background' class. Each object class has a number of images ranging from 31 to 800, and most classes have about 50 images. Our experiment setting is the same as in the original paper [4] and other papers, *e.g*., [5] and [16]. We divide the whole dataset into different numbers of training images per class, including 5, 15, 30, and each class has no more than 50 images for test. The image size is adjusted to be no larger than 300×300 pixels with preserved aspect ratio.

Caltech-256: The Caltech-256 dataset [5] comprises 30,607 images in 257 categories, *i.e*. 256 object classes and a 'clutter' class. The number of images per category is at least 80. The dataset exhibits larger varieties in size, location, and pose than in Caltech-101. As the Caltech-101 setting, we divide the whole dataset into a different number of training images per class, including 15, 30, 60, and each class has no more than 25 images for test.

Table 1. Number of sampled descriptors per class vs. mAP (%)

# of samples	10	25	50	75
RDF mAP (%)	69.44	71.09	73.52	73.59

Table 2. Number of trees vs. mAP (%)

# of trees	1	3	5	10	15
RDF mAP (%)	69.02	71.79	72.56	73.52	73.33

In all experiments, we use a dictionary of 10,000 visual words. The parameters of RDF include the number of trees, the tree height, and the size of a leaf node. We use Caltech-101 to analyze the relation between performance and the parameters. We find that the tree should not be too small, and so we set the height of tree from 11 to 13 and the stopping size of leaf node to be 5. In Table 1, we show the number of sampled descriptors of each class for building the RDF trees. It seems that 50 samples per class is a suitable number. That means, for Caltech-101 (102 classes), we use 5,100 descriptors to build a tree. On the other hand, Table 2 shows that the number of trees is also an important factor. As can been seen, when the number of trees is 15, the performance does not improve accordingly. A possible reason is that we only have 10,000 visual words in our dictionary. Each tree may take about 1,000 visual words, and as a result, the additional trees do not incorporate further information. We expect that, if we have more visual words, we can increase the diversity of trees and thus improve the

Table 3. Fifteen Scenes, Caltech-101, and Caltech-256 mAP (%). Other results are included if they are available in the original papers.

	Fifteen Scenes	Caltech-101			Caltech-256		
# of traning data per class	100	5	15	30	15	30	60
NBNN [1]	-	-	65.00	70.40	30.50	37.00	-
Local NBNN [10]	-	-	66.10	71.90	33.50	40.10	-
NBNN kernel [13]	-	-	61.30	69.60	-	-	-
SPM [6]	81.40	-	56.40	64.6	-	-	-
ScSPM [15]	80.28	-	67.00	73.20	27.73	34.02	40.14
LLC [14]	75.27	51.15	65.43	73.44	34.36	41.19	47.68
SVM-KNN [16]	-	46.60	59.08	66.20	-	-	-
Griffn-SPM [5]	-	44.20	59.00	67.60	28.30	34.10	-
RDT (RDF with only one tree)	76.00	48.81	62.39	69.02	28.37	33.75	38.71
RDF	82.12	51.49	65.91	73.52	33.73	40.91	45.63

performance. For the experiments on the Fifteen Scenes dataset and Caltech-101, we build an RDF of 10 trees. For Caltech-256, we build an RDF of only 6 trees.

To evaluate the performance of RDF, we repeat the experiments 10 times with randomly partitioned training and test sets on each of the three datasets, and calculate the average of accuracy per class in each run. Finally we compute the mean average precision (mAP) over the 10 runs. Our results are summarized in Table 3. We also include the corresponding results of other methods from the original papers if available. It can be seen that our method based on RDF can achieve better performance than most of the state-of-the-art methods. The computation time of RDF encoding is 1.3 times faster than LLC and 7 times faster than ScSPM. RDF encoding is fast because we only need to traverse trees and do not have to solve optimization problems.

4. Conclusion

The proposed RDF model has several advantages: *i*) The computation cost for randomized tree decomposition is low, and the tree/forest structure may facilitate parallel processing. *ii*) Each tree implies a specific way of decomposition. A descriptor would thus be represented by multiple sets of visual words, which is useful for improving the representation power. *iii*) The coupling of visual-word decomposition and random forest provides a seamless integration of unsupervised and supervised feature learning. We have shown that, despite its simplicity, the RDF model is quite effective and can achieve state-of-the-art performance on solving image categorization tasks.

Acknowledgment. This work is supported by NSC grant 101-2628-E-007-020-MY3.

References

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

[2] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *CVPR (1)*, pages 886–893, 2005.

[3] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin. LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874, 2008.

[4] L. Fei-Fei, R. Fergus, and P. Perona. Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. In *CVPRW'04*, pages 178–, 2004.

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

[6] S. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *CVPR (2)*, pages 2169–2178, 2006.

[7] F.-F. Li and P. Perona. A bayesian hierarchical model for learning natural scene categories. In *CVPR (2)*, pages 524–531, 2005.

[8] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 60(2):91–110, 2004.

[9] J. Mairal, F. Bach, J. Ponce, and G. Sapiro. Online dictionary learning for sparse coding. In *ICML*, page 87, 2009.

[10] S. McCann and D. G. Lowe. Local naive bayes nearest neighbor for image classification. In *CVPR*, pages 3650–3656, 2012.

[11] F. Moosmann, B. Triggs, and F. Jurie. Fast discriminative visual codebooks using randomized clustering forests. In *NIPS*, pages 985–992, 2006.

[12] D. Nistér and H. Stewénius. Scalable recognition with a vocabulary tree. In *CVPR (2)*, pages 2161–2168, 2006.

[13] T. Tuytelaars, M. Fritz, K. Saenko, and T. Darrell. The nbnn kernel. In *ICCV*, pages 1824–1831, 2011.

[14] J. Wang, J. Yang, K. Yu, F. Lv, T. S. Huang, and Y. Gong. Locality-constrained linear coding for image classification. In *CVPR*, pages 3360–3367, 2010.

[15] J. Yang, K. Yu, Y. Gong, and T. S. Huang. Linear spatial pyramid matching using sparse coding for image classification. In *CVPR*, pages 1794–1801, 2009.

[16] H. Zhang, A. C. Berg, M. Maire, and J. Malik. Svm-knn: Discriminative nearest neighbor classification for visual category recognition. In *CVPR (2)*, pages 2126–2136, 2006.