Integrated Feature Selection and Higher-order Spatial Feature Extraction for Object Categorization

David Liu[1], Gang Hua[2], Paul Viola[2], Tsuhan Chen[1]
Dept. of ECE, Carnegie Mellon University[1] and Microsoft Live Labs[2]
dliu@cmu.edu, {ganghua,viola}@microsoft.com, tsuhan@cmu.edu

Abstract

In computer vision, the bag-of-visual words image representation has been shown to yield good results. Recent work has shown that modeling the spatial relationship between visual words further improves performance. Previous work extracts higher-order spatial features exhaustively. However, these spatial features are expensive to compute. We propose a novel method that simultaneously performs feature selection and feature extraction. Higher-order spatial features are progressively *extracted based on selected lower order ones, thereby avoiding exhaustive computation. The method can be based on any additive feature selection algorithm such as boosting. Experimental results show that the method is computationally much more efficient than previous approaches, without sacrificing accuracy.*

1. Introduction

The traditional pipeline of pattern recognition systems consists of three stages: feature extraction, feature selection, and classification. These stages are normally conducted in independent steps, lacking an integrated approach. The issues are as follows: 1. **Speed**: Feature extraction can be time consuming. Features that require extensive computation should be generated only when needed. 2. **Storage**: Extracting all features before selecting them can be cumbersome when they don't fit into the random access memory.

Many object recognition problems involve a prohibitively large number of features. It is not uncommon that computing the features is the bottleneck of the whole pipeline. Techniques such as "classifier cascade" [17] reduce the amount of computation for feature extraction in run time (in testing), while the aim here is to improve the feature extraction and selection procedure in training.

In this work, we focus on the *bag-of-local feature descriptors* image representation [3] and its recent extensions [15][10][18]. Local feature descriptors are image statistics extracted from pixel neighborhoods or patches. Recent work of [15][10][18] focused on modeling the spatial relationship between pixels or patches. We call the features originated from local feature descriptors as 1^{st} *order features*, and features that encode spatial relationship between a set of two, three, or N patches as 2^{nd}, 3^{rd}, or N^{th} order features, respectively. Features with order larger than one are called *higher-order features*. These are analogous to *N-grams* [2] used in statistical language modeling. It is worth mentioning that, by higher-order features, we do **not** mean algebraic expansions (monomials) of lower order ones,

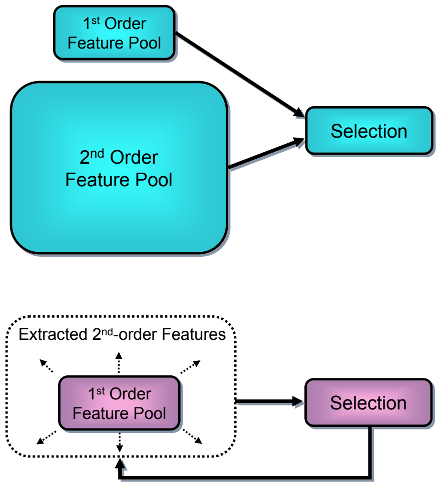


Figure 1. The top figure shows the traditional approach where 1^{st} and 2^{nd} order features are extracted before feature selection. In this paper, 2^{nd} order features encode spatial configurations of visual words and are expensive in terms of computation and storage. The proposal is to extract 2^{nd} order features based on previously selected 1^{st} order features and to progressively add them into the feature pool.

such as cross terms (x_1x_2), squares or cubes (x_1^3).

In the recent works of [15][10][18], higher-order features are extracted *exhaustively*. However, these higher-order features are prohibitively expensive to compute: first, their number is combinatorially exploding with the number of pixels or patches; second, extracting them requires expensive nearest neighbor or distance computations in image space [4]. It is the expensive nature of higher-order features that motivates our work.

Instead of *exhaustively* extracting all higher-order features before feature selection begins, we propose to extract them *progressively* during feature selection, as illustrated in Fig. 1. We start the feature selection process as early as when the feature pool consists only of 1^{st} order features. Subsequently, features that have been selected are used to create higher-order features. This process dynamically enlarges the feature pool in a greedy fashion so that we don't need to exhaustively compute and store all higher-order features.

A comprehensive review of feature selection methods is given by [8]. Our method can be based on any additive feature selection algorithm such as boosting [20] or CMIM [7][16]. Boosting was originally proposed as a classifier and has also been used as a feature selection method [17] due to its good performance, simplicity in implementation, and ease of extension to multiclass problems [20]. Another popular branch of feature selection methods is based on information-theoretic criteria such as maximization of conditional mutual information [7][16].

2. Integrated feature selection and extraction

Each image is represented as a feature vector which dynamically increases in the number of dimensions. Initially, each feature corresponds to a distinct codeword. The feature values are the normalized histogram bin counts of the visual words. These features are the 1^{st} order features, and this is the bag-of-visual words image representation [3]. Visual words, with textons [9] as a special case, have been used in various applications. A dictionary of codewords refers to the clusters of local feature descriptors extracted from pixel neighborhoods or patches, and a visual word refers to an instance of a codeword.

Our method maintains a 'feature pool' which initially consists only of 1^{st} order features. Subsequently, instead of exhaustively building all higher-order features, the process of *feature selection* and *higher-order feature extraction* are run alternately. At each round, *feature selection* picks a feature, and *feature extraction* pairs this feature with each of the previously selected features. The pairing process can be generic, and we will explain the implementation in Sec. 3. The pairing process creates new features which are concatenated to the feature vector of each image. In the next round of feature selection, this enlarged 'feature pool' provides the features to be selected from.

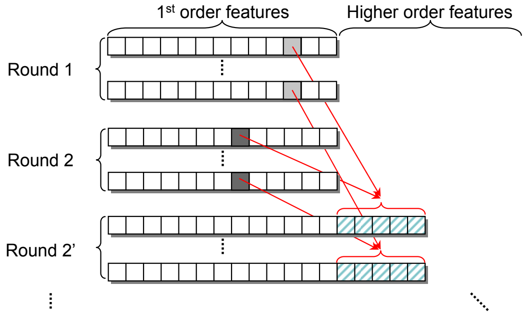


Figure 2. The 'feature pool' is dynamically built by alternating between feature selection and feature extraction.

In Fig. 2, we illustrate this process for the first few rounds. In the first round, *feature selection* picks a feature (the light gray squares) from the 'feature pool' and puts it in a 1^{st} order list (not shown in Fig. 2) that holds all previous selected 1^{st} order features. Since the list was empty, we continue to the second round. In the second round, *feature selection* picks a feature (the dark gray squares) from the 'feature pool' and places it in the 1^{st} order list. At the same time, *feature extraction* pairs this newly selected feature with the previously selected feature (the light gray square) and creates new features (the diagonally patterned squares). These 2^{nd} order features are then augmented into the 'feature pool'. In general, we may maintain $1^{st}, ..., L^{th}$-order lists instead of only 1^{st} order lists. If a selected feature has order L_1, then it was originated from L_1 codewords, and pairing it with another feature of order L_2 means that we can create new features that originate from a set of $L_1 + L_2$ codewords.

In Algorithm 1 we detail the procedure of computing features up to the 2^{nd} order. We use Discrete AdaBoost with decision stumps for feature selection as in [17], although other feature selection methods could be used as well. AdaBoost maintains a set of sample weights, $\{v_n\}, n = 1, ..., N$, on the N training images (Line 1). At each round, a decision stump tries to minimize the weighted error rate by picking an optimal feature and threshold (Line 4). The selected feature could be a 1^{st} or 2^{nd} order feature. If it is a 1^{st} order feature, it is placed in the 1^{st}-order list $z(.)$ (Line 8), and then paired with all previous members in the 1^{st}-order list to generate new 2^{nd} order features (Line 11). The new

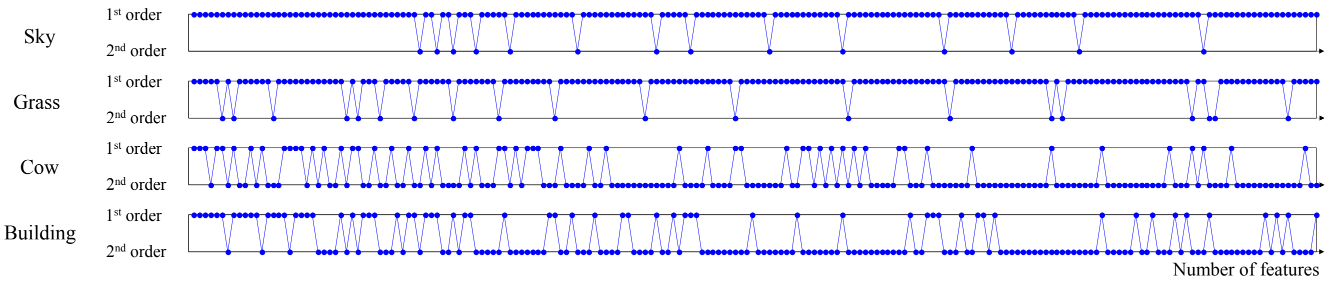


Figure 3. The order (1^{st} vs 2^{nd}) of a selected feature in each round.

Algorithm 1: Integrated-Feature-Selection-And-Spatial-Feature-Extraction

```
1  Sample weights $v_n^{(1)} \leftarrow 1/N$, $n = 1, ..., N$.
2  $k \leftarrow 0$.
3  for $m$=1,...,$M$ do
4      Fit decision stump $y_m(\mathbf{x})$ to training data by
       minimizing weighted error function
       $J_m = \sum_{n=1}^{N} v_n^{(m)} I(y_m(\mathbf{x}_n) \neq t_n)$
5      Denote feature index selected by decision
       stump as $i(m)$
6      if $i(m)$ corresponds to a $1^{st}$ order feature
       then
7          $k \leftarrow k + 1$
8          $z(k) \leftarrow i(m)$
9          for $j$=1,...,$k$-1 do
10             for each image do
11                 BuildSecondOrderFeatures($z(k), z(j)$)
12             end
13             Augment feature pool
14         end
15     end
16     $\epsilon_m \leftarrow \frac{\sum_{n=1}^{N} v_n^{(m)} I(y_m(\mathbf{x}_n) \neq t_n)}{\sum_{n=1}^{N} v_n^{(m)}}$ and $\alpha_n \leftarrow \ln \frac{1-\epsilon_m}{\epsilon_m}$
17     $v_n^{(m+1)} \leftarrow v_n^{(m)} \exp\{\alpha_n I(y_m(\mathbf{x}_n))\}$
18 end
19 Selected features are $\{\mathbf{x}_{i(1)}, ..., \mathbf{x}_{i(M)}\}$ for any
   vector $\mathbf{x}$
```

features are augmented into the feature pool (Line 13).

Lines 16 and 17 are standard update rules of AdaBoost. It updates the sample weights in a manner so that the decision stumps can focus on the source of error. This eventually drives the choice of features. Using AdaBoost as a feature selection tool is justified by its taking into account the classification error when selecting features [7]. However, the concept of integrating feature selection and extraction is general, and the feature extraction procedure in lines 6 to 15 can be embedded into other feature selection methods as well.

To show that different object categories result in different temporal behaviors of the integrated feature selection and extraction process, we show in Fig. 3 the order of a selected feature at each round of boosting, from rounds 1 to 200. AdaBoost is used in a binary one-vs-rest classification manner. In the first few rounds, 1^{st} order features are being selected and 2^{nd} order features are being built. Structured objects such as 'Cow' and 'Building' soon start to select 2^{nd} order features. At the end, structured objects tend to select more 2^{nd} order features compared to homogeneous objects such as 'Sky'. This agrees with the expectation that sky has less obvious geometrical structure between pairs of 1^{st} order features.

After feature selection and extraction, to make predictions, one can:
1. treat boosting solely as a feature selection tool and use the selected features, $\{\mathbf{x}^{i(1)}, ..., \mathbf{x}^{i(M)}\}$, as input to any classifier; or,
2. proceed as in AdaBoost and use a thresholded weighted sum, $Y(\mathbf{x}) = \text{sign}(\sum_{m=1}^{M} \alpha_m y_m(\mathbf{x}))$, as the final classifier; or,
3. as we propose, use the set of weighted decision stumps, $\{\alpha_1 y_1(\mathbf{x}), ..., \alpha_M y_M(\mathbf{x})\}$, as features and train a linear SVM.
We will experiment with the last two methods later.

3. Second-order spatial features

The algorithm introduced in the previous section is a generic method for integrating the feature selection and feature extraction processes. In this section we provide examples of building 2^{nd} order features, given a pair of 1^{st} order features, (w_a, w_b) (Line 11 in Algorithm 1). In the Experiments section, we will explain how 3^{rd} order features can be built.

Different kinds of spatial histograms can be used for building 2^{nd} order features. In Fig. 4(a), we illustrate a spatial histogram with distance approximately in log

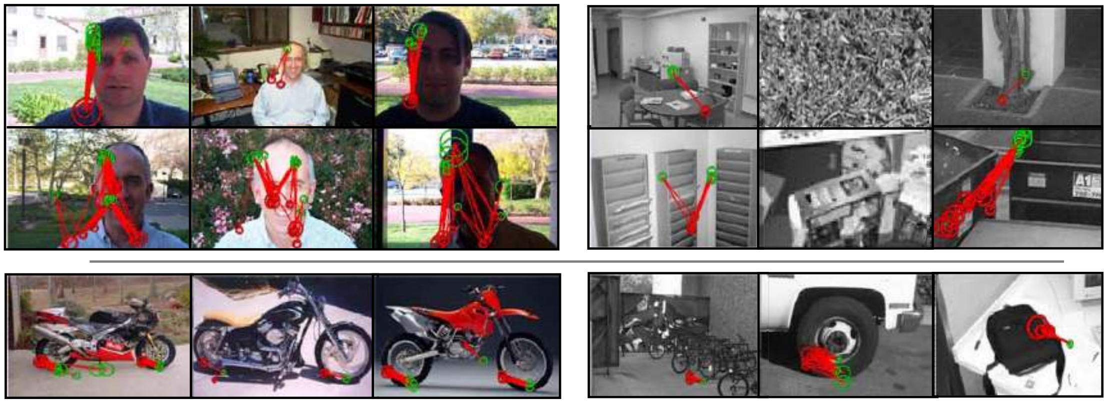


Figure 5. Second-order features. These are best viewed in color.

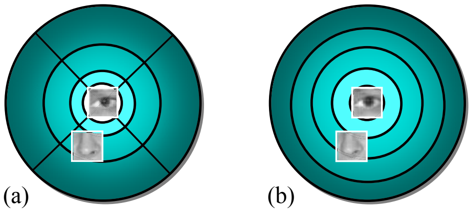


Figure 4. Examples of spatial histograms.

scale, similar to the shape context histogram [1]. The log scale tolerates larger uncertainties of bin counts in longer ranges. The four directional bins are constructed to describe the semantics 'above', 'below', 'to the left', and 'to the right'. In Fig. 4(b), directions are ignored in order to describe how the co-occurrence of (w_a, w_b) varies in distance. In [15], squared regions are used to approximate the circular regions in Fig. 4(b) in order to take advantage of the integral histogram method [14]. Of course, squared regions and integral histogram can be used in our work as well.

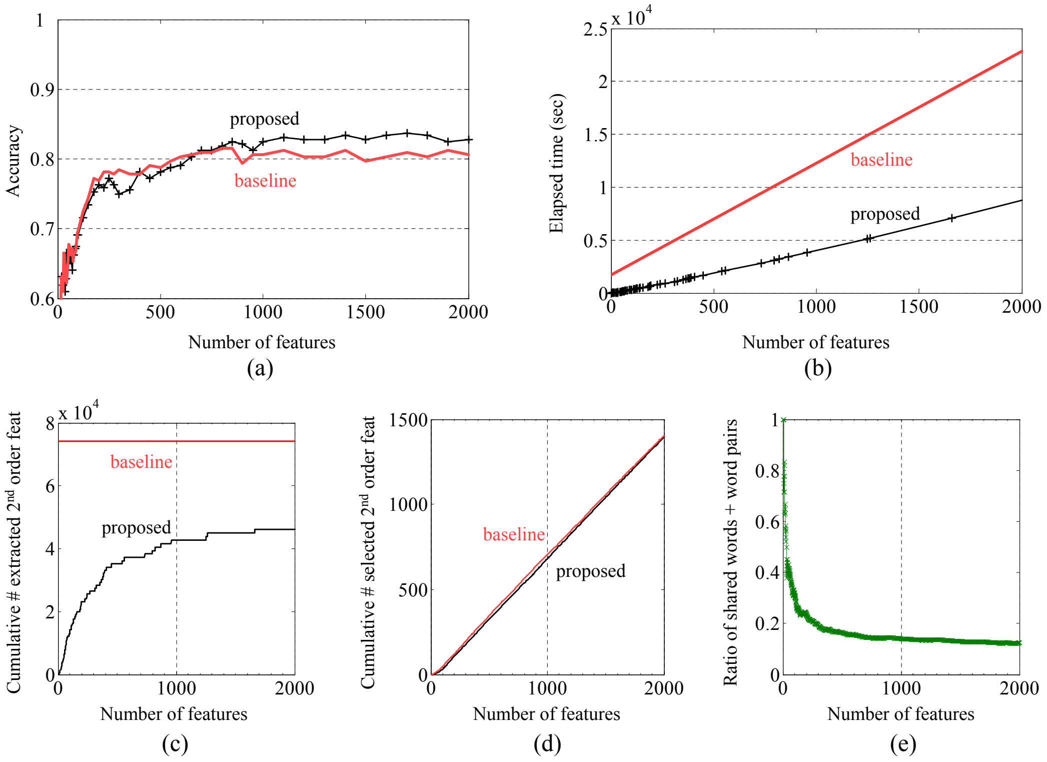
The goal is to build a descriptor that describes how w_b is spatially distributed relative to w_a. Let us first suppose that there is only a single instance of w_a in an image, but multiple w_b's. Using this instance of w_a as a reference center of the spatial histogram, we count how many instances of w_b fall into each bin. The bin counts form the descriptor. Since there are usually multiple instances of w_a in an image, we build a spatial histogram for each instance of w_a, and then normalize over all spatial histograms; the normalization is done by summing the counts of corresponding bins, and dividing the counts by the number of instances of w_a. This takes care of the case when multiple instances of an object appear in an image. The whole process is summarized in Algorithm 2.

The spatial histograms yield translation invariant descriptors, since the reference center is always in respect to the center word w_a, and describes the relative position of instances of w_b. The descriptors can also be (quasi-)scale invariant. This can be achieved by determining the normalized distance between instances of w_a and w_b, where the normalization is done by considering the geometric mean of the scale of the two patches. To make the descriptor in Fig. 4(a) rotation invariant, we can take into account the dominant orientation of a patch [19]. However, rotation invariance may diminish discriminative power and hurt performance [19] in object categorization.

Algorithm 2: BuildSecondOrderFeatures

```
Goal: create feature descriptor given a word pair
Input: codeword pair $(w_a, w_b)$
Output: a vector of bin counts
Suppose there are $N_a$ instances of $w_a$, and $N_b$
  instances of $w_b$ in the image
Initialize $N_a$ spatial histograms, using each
  instance of $w_a$ as a reference center
for $i$=1,...,$N_a$ do
    Count the number of instances of $w_b$ falling in
    each bin
end
Sum up corresponding bins over the $N_a$ spatial
  histograms
Divide bin counts by $N_a$
```

In Fig. 5, red circles indicate words used as reference center. The red-green pairs correspond to a highly discriminative 2^{nd} order feature that has been selected in early rounds of boosting. The images are those that are incorrectly classified when only 1^{st} order features are used for training a classifier. We can see that 2^{nd} order features can detect meaningful patterns in these images. As a result, most of these images are correctly



(a)

(b)

(c)

(d)

(e)

Figure 6. Integrated vs separated: After around 800 rounds of boosting, the proposed method outperforms baseline both in (a) testing accuracy and (b) required training time.

classified by a classifier using both 1^{st} and 2^{nd} order features.

4. Experiments

We use three datasets in the experiments: the PASCAL VOC2006 dataset [5], the Caltech-4 plus background dataset used in [6], and the MSRC-v2 15-class dataset used in [15]. We used the same training-testing experiment setups as in these respective references.

For each dataset we use different local feature descriptors to show the generality of our approach. For the PASCAL dataset, we adopt the popular choice of finding a set of salient image regions using the Harris-Laplace interest point detectors [5]. Another scheme is to abandon the use of interest point detectors [13] and sample image patches uniformly from the image. We adopt this approach for the Caltech-4 dataset. Each region or patch is then converted into a 128-D SIFT [12] descriptor. For the MSRC dataset, we follow the common approach [15] of computing dense filter-bank (3 Gaussians, 4 Laplacian of Gaussians, 4 first order derivatives of Gaussians) responses for each pixel.

The local feature descriptors are then collected from the training images and vector quantized using K-means clustering. The resulting cluster centers form the dictionary of codewords, $\{w_1, ..., w_J\}$. We use $J = 100$ for the MSRC dataset, and $J = 1000$ for the other two datasets; these are common choices for these datasets. Each local feature descriptor is then assigned to the closest codeword and forms a visual word.

For the MSRC dataset, we used the spatial histogram in Fig. 4(b), in order to facilitate comparison with the recent work of [15]. We followed the specs in [15] with 15 distance bins of equal spacing, the outermost bin with a radius of 80 pixels, and no scale normalization being performed. For the Caltech and PASCAL datasets, we used the spatial histogram in Fig. 4(a), where the scale is normalized according to the patch size or interest point size as explained earlier, and the outermost bin has a radius equal to 15 times the normalized patch size. The scale invariance can be observed in Fig. 5 from the different distances between red-green word pairs.

4.1. Integrated vs Separated

Here we present the **main result** of this paper. In Fig. 6 we show the experiment on the 15-class MSRC dataset. We use a multiclass version of AdaBoost [20] for feature selection, and linear SVM for classification as explained in Sec. 2. In Fig. 6(a), we see that the accuracy settles down after about 800 rounds of boosting. Accuracy is calculated as the mean over the diagonal elements of the 15-class confusion matrix. In Fig. 6(b), we see the integrated feature selection and extraction scheme requires only about 33% of training time compared to the *canonical* approach where feature extraction and selection are two *separate* processes.

Surprisingly, we can see in Fig. 6(a) that, in addition to being more efficient, the proposed scheme also achieves better accuracy in spite of its greedy nature. This can be explained by the fact that 2^{nd} order features are sparser than 1^{st} order features and hence statistically less reliable; the integrated scheme starts with the pool of first order features and gradually adds in 2^{nd} order features, hence it spends more quality time with more reliable 1^{st} order features.

In Fig. 6(c)-(e) we examine some temporal behaviors of the two methods. In Fig. 6(c), we show the cumulative number of 2^{nd} order features being extracted at each round of feature selection. While the canonical procedure extracts all features before selection starts, the proposed scheme aggressively extracts 2^{nd} order features in earlier rounds and then slows down. This logarithmic type of curve signifies the coupling between the feature extraction and the feature selection processes; if they weren't coupled, features would have been extracted at a constant (linear) speed instead of a logarithmic.

In Fig. 6(c), we also noticed that at 800 rounds of boosting, only about half of all possible 2^{nd} order features were extracted. This implies less computation in terms of feature extraction, as well as more efficient feature selection, as the feature pool is much smaller.

In Fig. 6(d), it appears that the canonical approach selects 2^{nd} order features at roughly the same pace as the integrated scheme, both selecting on average 0.7 second-order features per round of boosting. But in fact, as shown in Fig. 6(e), the overlap between the selected features of the two methods is small; at 800 rounds of boosting, the share ratio is only 0.14. The share ratio is the intersection of the shared visual words and visual word pairs of the two methods divided by the union. This means that the two methods have very different temporal behaviors.

4.2. Importance of feature selection

Here we compare with the recent work of [15], where feature selection is not performed, but first and second-order features are quantized separately into dictionaries of codewords. A histogram of these codewords is used as a feature vector. In Table 1, all three methods use the nearest neighbor classifier as in [15] for fair comparison [1]. We see that our method yields state-of-the-art performance, compared to the quantized (Method 2) and non-quantized (Method 1) versions. In addition, since the 2^{nd} order features need not be exhaustively computed and also no vector quantization on 2^{nd} order features is required, our method is also much faster than the method in [15].

	Proposed	Method 1	Method 2 [15]
Feature selection	√	×	×
Quantization	×	×	√
Accuracy	75.9%	71.3%	74.1%

Table 1. Importance of feature selection.

4.3. Linear SVM on weighted decision stumps

As explained in Sec. 2, we propose to concatenate the weighted output of all weak classifiers, $\{\alpha_1 y_1(\mathbf{x}), ..., \alpha_M y_M(\mathbf{x})\}$, from AdaBoost as a feature vector and then run a linear SVM. Results are shown in Table 2. The superior result over AdaBoost comes from a re-weighting of the terms $\{\alpha_1 y_1(\mathbf{x}), ..., \alpha_M y_M(\mathbf{x})\}$.

	PASCAL (EER)	MSRC (1-accuracy)
AdaBoost classifier (1^{st} order feat)	13.4%	24.1%
AdaBoost classifier (1^{st} & 2^{nd} order)	12.1%	21.2%
Linear SVM on weighted decision stumps	10.9%	16.9%

Table 2. Performance on the PASCAL car-vs-rest and MSRC 15-class datasets.

The best results [5] reported on the PASCAL VOC2006 and VOC2007 datasets employ the Spatial Pyramid [11] technique on top of the bag of words representation. The Spatial Pyramid technique is orthogonal to the proposed method and combining them is expected to yield even better results.

4.4. Increasing the order

In Fig. 7, we experiment on the MSRC dataset and see that the classification accuracy obtained from using a feature pool of 1^{st} and 2^{nd} order features is higher than using 1^{st} order features alone. Including 3^{rd} order features does not improve accuracy. We generated 3^{rd} order features by counting the number of times three codewords (w_a, w_b, w_c) fall within a radius of 30 pixels, i.e., the spatial histogram has only one bin. Third

[1] We re-implemented the work of [15], because they used an untypical quantization scheme to generate 1^{st} order codewords, and results are not comparable; also, their spatial histogram is square-shaped.

order features are generated every time a 1^{st} order feature is selected (which corresponds to w_a) and paired with each of the previously selected 2^{nd} order features (recall that a 2^{nd} order feature comes from a word pair, (w_b, w_c)), or vice versa. The reason for reducing the number of bins to one is to account for the data sparseness of higher-order features, which we will discuss later.

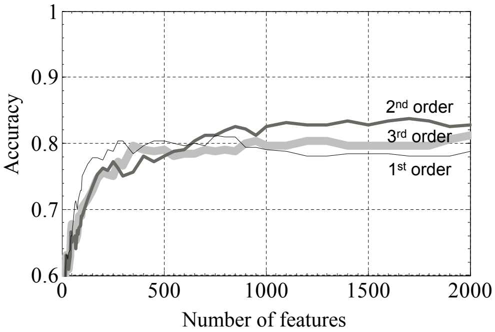


Figure 7. Accuracy and feature complexity.

4.5. Robustness of co-occurrence counts

Instead of assigning a local feature descriptor to a single codeword, one can assign it to the top-N closest codewords. In Table 3, we vary the parameter c_1 from one to four and ten, which is the number of codewords each image patch is assigned to. In three out of four categories, the performance of the bag of words representation (using 1^{st} order features only) degrades as c_1 increases from one to four or ten, which manifests the popular practice of assigning a descriptor to a single codeword.

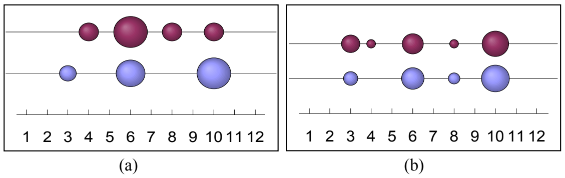


Figure 8. Effect of parameter c_2 on the spatial histogram bin counts. (a) Using $c_2 = 1$. (b) Using $c_2 = 10$.

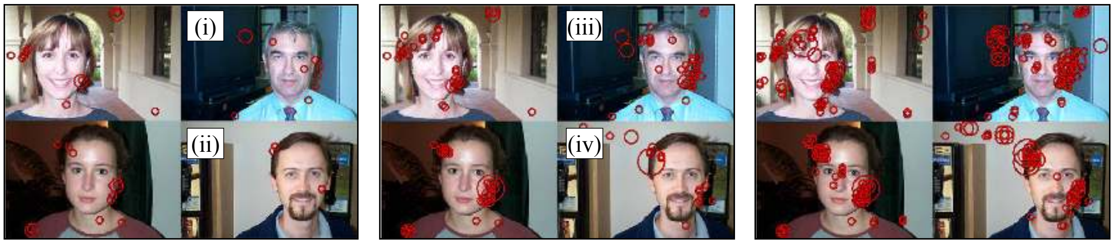
Yet, the top-N technique can help avoid the data-sparseness problem of 2^{nd} order features. We define the parameter c_2 as the number of visual words each image patch is assigned to when constructing 2^{nd} order features. Notice that c_1 and c_2 can have different values. In Fig. 8 we show the benefit of increasing c_2 from one to ten when constructing spatial features. In Fig. 8(a), two normalized spatial histograms with twelve spatial bins are collected from two different face images. The size of the bubbles indicates normalized bin counts. Recall that spatial histograms collect spatial co-occurrence of word pairs; in this case the specific word pair corresponds to a person's nose and eye from real data. Ideally the two histograms would be nearly identical, but image variations and clustering artifacts prevent it from being so. In Fig. 8(b), using the top-N technique, the two histograms become more similar to each other. The reason that 2^{nd} order features benefit more from this technique than 1^{st} order ones is due to the sparsity of co-occurrence of a word pair. The chance of *co-occurrence* between a pair of visual words within a specific spatial bin is at the order of approximately $1/(J^2 \times 12)$, where J is the size of the dictionary of codewords. Compared to the order of $1/J$ for the histogram of visual words, slight image variations and clustering artifacts can result in larger disturbances in the spatial feature bin counts than in the visual word bin counts. The top-N technique increases the bin counts (before normalization) and reduces the sensitivity to variations. In Fig. 9 we see the population of a particular codeword getting denser as c_2 increases. In Fig. 9(i)(ii), this codeword rarely appears 'correctly' on the chin of the face. Increasing c_2 increases its occurrence on the chin, but also increases its occurrence at other locations, so increasing c_2 indefinitely would lead to performance degrading. Overall, this suggests that using a small value of c_1 but a moderate value of c_2 should give the best result. Indeed, using AdaBoost as classifier, we found that $(c_1 = 1, c_2 = 10)$ gives state-of-the-art performance, as shown in Table 3.

(c_1,c_2) Class		(1,1)	(4,4)	(10,10)	(1,10)
Face	1^{st} order feat	4.15	3.23	5.53	4.15
	1^{st} and 2^{nd} order feat	1.84	1.84	0.92	0.92
Motorbike	1^{st} order feat	1.50	2.00	2.75	1.50
	1^{st} and 2^{nd} order feat	1.50	1.25	1.00	1.00
Airplane	1^{st} order feat	2.75	4.00	4.00	2.75
	1^{st} and 2^{nd} order feat	2.25	2.50	2.00	1.75
Car	1^{st} order feat	1.00	1.50	2.25	1.00
	1^{st} and 2^{nd} order feat	0.50	0.75	1.00	0.50

Table 3. Equal error rates (%) for the Caltech-4 dataset. By integrating feature selection and extraction, state-of-the-art results are obtained.

5. Conclusion and future work

We have presented an approach for integrating the process of feature selection and feature extraction. The integrated approach is three times faster than



(a) Top 1 (b) Top 4 (c) Top 10

Figure 9. Effect of increasing the number of visual words a patch is assigned to.

the canonical procedures of feature selection followed by feature extraction. In addition, the integrated approach can achieve comparable or even better accuracy than the exhaustive approach, in spite of its greedy nature.

Our approach is generic and can be used with other feature selection methods. It can also be applied to all kinds of spatial histograms. In this work, we considered non-parametric histograms (with spatial bins), but parametric ones could be used as well, where the parameters (e.g., the mean and covariance of point clouds) could be used as features.

Finally, we presented detailed experiments on three different object categorization datasets which have been widely studied. These datasets cover a wide range of variations on object category (20 in total), object scale (most noticeably in the PASCAL dataset) and pose. For each dataset, we used different state-of-the-art local feature descriptors. These experiments demonstrate that our approach applies to a wide range of conditions.

References

[1] S. Belongie, J. Malik, and J. Puzicha. Shape matching and object recognition using shape contexts. *PAMI*, 24:509–522, 2002.

[2] P. Brown, V. Della Pietra, P. de Souza, J. Lai, and R. Mercer. Class-based n-gram models of natural language. *Comp. Linguistics*, 18(4):467–479, 1992.

[3] G. Csurka, C. Dance, L. Fan, J. Willamowski, and C. Bray. Visual categorization with bags of keypoints. In *ECCV Workshop Statistical Learning*, 2004.

[4] M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf. *Computational Geometry, Chap. 5.* Springer-Verlag, second edition, 2000.

[5] M. Everingham, A. Zisserman, C. K. I. Williams, and L. Van Gool. PASCAL VOC2006 Results. http://www.pascal-network.org/challenges/VOC/voc2006.

[6] R. Fergus, P. Perona, and A. Zisserman. Object class recognition by unsupervised scale-invariant learning. *CVPR*, 2003.

[7] F. Fleuret. Fast binary feature selection with conditional mutual information. *JMLR*, 5:1531–1555, 2004.

[8] I. Guyon and A. Elisseeff. An introduction to variable and feature selection. *JMLR*, 3:1157–1182, 2003.

[9] B. Julesz. Textons, the elements of texture perception and their interactions. *Nature*, 290:91–97, 1981.

[10] X. Lan, C. L. Zitnick, and R. Szeliski. Local bi-gram model for object recognition. Technical report, MSR-TR-2007-54, Microsoft Research, 2007.

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

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

[13] E. Nowak, F. Jurie, and B. Triggs. Sampling strategies for bag-of-features image classification. In *ECCV*, 2006.

[14] F. Porikli. Integral histogram: a fast way to extract histograms in cartesian spaces. *CVPR*, 2005.

[15] S. Savarese, J. Winn, and A. Criminisi. Discriminative object class models of appearance and shape by correlatons. *CVPR*, 2006.

[16] M. Vidal-Naquet and S. Ullman. Object recognition with informative features and linear classification. *ICCV*, 2003.

[17] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. *CVPR*, 2001.

[18] L. Yang, P. Meer, and D. Foran. Multiple class segmentation using a unified framework over mean-shift patches. *CVPR*, 2007.

[19] J. Zhang, M. Marszalek, S. Lazebnik, and C. Schmid. Local features and kernels for classification of texture and object categories: A comprehensive study. *IJCV*, 73 (2):213–238, 2007.

[20] J. Zhu, H. Zou, S. Rosset, and T. Hastie. Multi-class adaboost. *Submitted*, 2005.