Efficient Kernels for Identifying Unbounded-Order Spatial Features

Yimeng Zhang
Carnegie Mellon University
yimengz@andrew.cmu.edu

Tsuhan Chen
Cornell University
tsuhan@ece.cornell.edu

Abstract

Higher order spatial features, such as doublets or triplets have been used to incorporate spatial information into the bag-of-local-features model. Due to computational limits, researchers have only been using features up to the 3rd order, i.e., triplets, since the number of features increases exponentially with the order. We propose an algorithm for identifying high-order spatial features efficiently. The algorithm directly evaluates the inner product of the feature vectors from two images to be compared, identifying all high-order features automatically. The algorithm hence serves as a kernel for any kernel-based learning algorithms. The algorithm is based on the idea that if a high-order spatial feature co-occurs in both images, the occurrence of the feature in one image would be a translation from the occurrence of the same feature in the other image. This enables us to compute the kernel in time that is linear to the number of local features in an image (same as the bag of local features approach), regardless of the order. Therefore, our algorithm does not limit the upper bound of the order as in previous work. The experiment results on the object categorization task show that high order features can be calculated efficiently and provide significant improvement in object categorization performance.

1. Introduction

The visual problems of generic object categorization and clustering are challenging. The model solving these problems needs to have enough discriminative power in order to differentiate objects from different categories. Besides, the computational complexity in terms of speed and storage are always important issues to address when we try to increase the effectiveness of the model. In this paper, we will propose a novel kernel algorithm which gains more discrimination with little computational cost.

Most recent works are based on the local features of the image. The “bag of features” representation [3][20] uses only the local appearance information from the image, and discards completely the geometry information. Therefore, this representation of the image is advantageous in computational complexity and invariance within category.

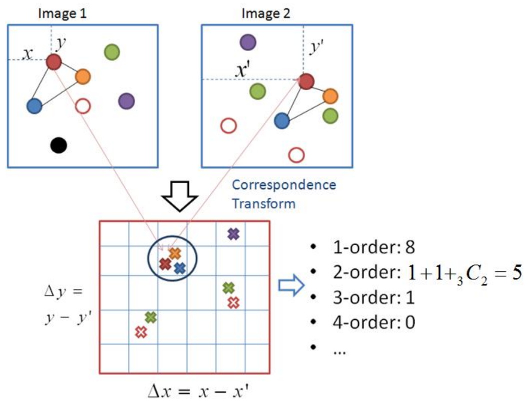


Figure 1: Illustration of the basic idea. Each circle in the top two images corresponds to a visual word (local feature). Different colors represent different words. Two images are transformed to the offset space (bottom image) in order to find the co-occurrence of high order features. Each cross in the offset space is created by a pair of same words (same color) form the two input images. The main idea is that when n points have the same location in the offset space, we have a particular co-occurring n order feature.

However, due to the fact that it does not model the spatial layout of the local features and therefore the shape of the objects, it loses some discriminative power, since most objects are either strictly or loosely structured.

Many works have been done to incorporate geometric information to the bag-of-features model, which will usually result in more computational complexity, exponential or polynomial to the number of features. Constellation models [16][4] represent the objects with a fixed number of parts which are composed with the local features, and capture the geometry information by modeling the spatial layout of the parts, usually with a joint Gaussian. This type of models is computational expensive in that it requires searching an exponentially large number of hypothesis which give different part assignments to the features. The second type is star shaped models [19][14], which exploits geometry information by modeling the locations of the local features relative to the center of the object. These models can be easily trained, while usually require searching for an optimal object center in the image during testing. Both constellation models and star shaped models require the training images with bounding boxes.

In this work, we focus on the third type of technique that uses the mutual geometric relationship between local features [18][11][10]. Higher order features are constructed with a specific number of local features together with their spatial layout. Following the definition from [11], we call the local features 1st order features, and features with two, three, n local features, 2nd, 3rd and n^{th} order features. One advantage with this type of models comparing to the previous two types is translation invariance, and therefore the models can handle supervised learning with training examples only labeled with their categories. However, as the order n increases, the number of features will immediately reach an intractable amount. Therefore, most previous work end at 2nd order features [18], or at most until 3rd order [11][10]. Moreover, in order to reduce the computation, higher order features are usually created with only local features lying within a certain distance, which makes the models unable to capture long range interactions in the image.

We propose an efficient algorithm which is capable of handling spatial high order features. The main idea is illustrated in Figure 1. In order to find the co-occurring high order spatial features between two images, we transform the local features to the offset space. A point is created in the offset space by two corresponding local features between the two images. With the visual word concept, corresponding local features are the features described by the same words. The location in the offset space is the relative location difference between the two words. After transforming to the offset space, the word pairs of a co-occurring high order feature will be at the same location in order to ensure the same spatial layout. After the transformation, it would be quite easy to find co-occurring high order features, which is intractable in the original image space.

We calculate the inner product of the feature vectors of n^{th} order features (n can be any large number) of two images, which can be used as a kernel function for any kernel based learning algorithms, such as SVM, or Kernel PCA. In standard procedure[11][18], the system first transforms the input images into feature vectors, and then calculates the distance among the feature vectors, both of which are computationally expensive with a large number of features, and prevent us from using higher order features. Borrowing the idea of the kernel methods, we calculate the inner product directly without first extracting the features for the two images. We show that given the method for finding co-occurring high order features, the inner product can be calculated in linear time to the number local features in an image, regardless of the order k or the distance between the local features. The computation efficiency makes us capable of handling long range interaction and any large order features.

Recent years, there have been works [8][7][9] that use link analysis techniques to implicitly model the high order features. These works first link among images by comparing the pair-wise features between images, and then they use link analysis to explore the links in order to find large connected clusters, which is similar to the concept of high order features. The advantage of our work comparing to their works is that we can compute high order features faster than theirs. They first compare pair-wise features, it is similar as a direct computation of 2nd order features, and for the link analysis, and usually an Eigen analysis needs to be done on a large matrix. Moreover, our method can be easily used as a kernel function, and thus we can take advantage of some powerful learning algorithms.

There have been many other kernels proposed for object categorization [5][1][10]. However, these kernels are either 1) not designed to capture spatial information [5], 2) not translation invariant [1] since they use absolute coordinates to capture the spatial information, or 3) computationally expensive [10]. Besides, none of these kernels are designed to calculate higher order (larger than 2) features. Our kernel addresses all these issues. High order kernels have been designed for many other applications, such as the string kernel [12] for document classification;

2. Kernel with High Order Features

In this section, we describe the kernel of two images with high order features. A kernel function is a function that calculates the inner production between two examples after mapping to the feature space. For any mapping $\phi: X \to F$, from the input space X, to the feature space F, a kernel function is as follows.

$$K(x_i, x_j) =< \phi(x_i), \phi(x_j) >, \forall x_i, x_j \in X \quad (1)$$

The idea is to compute the inner product by implicitly mapping the data to the high dimensional feature space.

We first describe the image representation in section 2.1, and then define the feature space F and the mapping ϕ in 2.2, and propose the algorithm that calculates the kernel in section 2.3. Moreover, we describe the solutions to some practical issues in section 2.4 and 2.5, and finally we discuss about the computation complexity of the algorithm in section 2.6.

2.1. Image Representation

An image I is represented as a collection of visual words w [3], and each word is associated with their location r on the image. $I = \{(w_1, r_1), (w_2, r_2), \dots (w_m, r_m)\}$. The locations can be detected with interest point and region detectors, and the words are defined by clustering the local features extracted at the interest regions.

2.2. Kernel with High Order Spatial Features

We define the features with one word 1st order features, and features with two, three, n words, 2nd, 3rd and nth order features. Different relative spatial distribution among the n visual words yields different n order features. The value of each feature in the image is the number of occurrences of that feature in the image.

Specifically, let Σ be the visual word vocabulary. An n^{th} order feature f_n would be n words from Σ with a specific relative spatial layout. In this section, we only consider translation variance about the features, and ignore any rotation, or scaling effect of the features, so the same feature must have the same pixel-wise spatial layout. We will deal with the quantization, scaling and rotation in section 2.3 and 2.4. The length of the feature vector would be exponential to n, approximately $|\Sigma|^n \times g_n (g_n$ is the number of possible special layout for n words). The f_n coordinate $\phi_{f_n}(x)$ of the n^{th} order feature vector $\phi_n(x)$ of an image x is defined as the number of occurrences of the feature in the image.

$$K_n(x,y) =< \phi_n(x), \phi_n(y) > = \sum_{f_n} < \phi_{f_n}(x), \phi_{f_n}(y) > \quad (2)$$

To remove the bias introduced by the number of visual words of an image, we normalize the feature vector as $\frac{\phi_n(x)}{\|\phi_n(x)\|}$. The kernel becomes as follows

$$\widehat{K}_n(x,y) = < \frac{\phi_n(x)}{\|\phi_n(x)\|}, \frac{\phi_n(y)}{\|\phi_n(y)\|} >= \frac{K_n(x,y)}{\sqrt{K_n(x,x)K_n(y,y)}}$$

Our final kernel is a weighted sum of all $\widehat{K}_n$'s.

$$K(x,y) = \sum_{n=1}^{\infty} w_n \widehat{K}_n(x,y) \quad (3)$$

Where the weights $w_n = \mu^{1-n}$, μ is between 0 and 1. As μ gets closer to zero, we put more weight to the higher order features.

Since we define the kernel as an inner product in the feature space, it satisfies the Mercer's condition (symmetric and positive semi-definite) from its definition.

A direct computation of $K_n(x,y)$ would require $O(|\Sigma|^n \times g_n)$ time and space. To make the computation efficient, first, we rewrite the kernel function in equation (2) as follows.

$$\begin{aligned} K_n(x,y) &= \sum_{f_n} < \phi_{f_n}(x), \phi_{f_n}(y) > \\ &= \sum_{\mathrm{f_n}} \sum_{\mathrm{u_n=f_n, u_n \in x}} 1 \times \sum_{\mathrm{u_n=f_n, u_n \in y}} 1 \\ &= \sum_{\mathrm{f_n}} \sum_{\mathrm{u_n=f_n, u_n \in x}} \sum_{\mathrm{u_n=f_n, u_n \in y}} 1 \quad (4) \end{aligned}$$

Here, $u_n \in x, u_n = f_n$ means that u_n is an instance of nth order feature f_n present in image x. Let $x = \{(w_1, r_1), (w_2, r_2), \dots (w_m, r_m)\}$, s is a subset of $\{1, .., m\}$ with size n. $u_n = \{(w_{s_1}, r_{s_1}), \dots, (w_{s_n}, r_{s_n})\}$.

Hence, although the length of the feature vectors ϕ_n of two images x and y can be quite large, the inner product of them is just the sum of the co-occurrence of all n^{th} order features. Our goal would be to count the features occurring in both images.

2.3. Correspondence Transform

It would be quite difficult to find all co-occurring spatial features in the image space, given the large dimension of the feature space and the large amount of 2D distributed visual words per image.

The main idea of the algorithm is that if u_n in image x, and v_n in image y are the same n^{th} order feature, then v_n must be a constant shift of the same visual words of u_n in the image space. Thus, we can simplify the task of counting co-occurrence of n^{th} order features in two images to counting the constant shift n visual word pairs in the two images. However doing this directly on the image space would be still too much.

In order to facilitate this process, we first transform the feature points in two images to the offset space. The idea is illustrated in Figure 1. We call this 'Correspondence Transform'. First, we pair all the same visual words in the two images (same color circles in the figure). Note that if for a word in one image, there are multiple correspondences in the other image, we create multiple pairs, as the green and red ring word in Figure 1. A pair can be represented as (w, r_1, r_2), where w is the visual word of the pair, r_1 is the location of the word in one image, and r_2 is the location in the other image. The number of pairs is in fact the inner product of the first order feature vector (bag of words). Then, we calculate the difference of the locations for each pair Δr, $\Delta r = r_1 - r_2$. This would become our offset space.

Since r is a vector of coordinates of x and y axies, Δr is also a vector.

$$\Delta r = (\Delta x, \Delta y) = (x_1 - x_2, y_1 - y_2)$$

Thus, the offset space is a 2D space as in the bottom image of Figure 1. Each pair from the two input images corresponds to one point on this space. Following the constant shift idea, if we have a co-occurring n order feature, the word pairs from this feature would fall to the same location on the offset space in order to ensure the relative spatial layout. For example, in Figure 1, the particular 3rd order feature with red, blue, and orange points co-occurs in the two images, therefore, the word pairs of them fall to the same location in the offset space. After the transformation, finding co-occurring high order features becomes a trivial task.

Now, we calculate the kernel values for all K_n of different n on the offset space. We cluster the pairs fall into the same location in the offset space. The value of the n^{th} order kernel function would be the number of clusters of n pairs with the same offset. Note that when we have a cluster of n pairs with the same offset, it also gives us ${}_iC_k$ number of clusters of i pairs with the same offset for any $i < n$. In the example of Figure 1, K_1 is the number of pairs 8, $K_2 = 1 + 1 + {}_3C_2$ since we have two clusters of size 2, and 1 cluster of size 3, $K_3 = 1$ for the cluster of three pairs in the same offset, and for all $n > 3$, $K_n = 0$ since we do not have any cluster of size larger than 3. We can easily verify

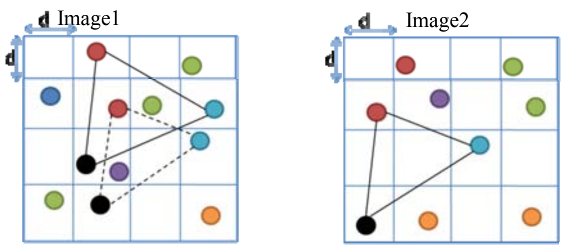


Figure 2: An example for spatial layout of the features. Different colors represent different words.

that this is exactly the same as we calculate the inner product of the feature vectors directly. We summarize the algorithm as in Algorithm 1. The kernel will be further normalized as in stated in section 2.2.

The algorithm does not give any constraint or size limit to the order n, since both the computation time and storage of it do not depend on n. With this algorithm, we are actually calculating the kernels of all order features (1 to infinity). Of course due to the size of the dictionary (codebook) and the images, the value of kernels with large order features would be 0. But algorithm itself does not limit the order.

Algorithm 1: Compute n^{th} Order Kernel for n = 1,2,…,∞

Input: Two images $\boldsymbol{I_1} = \{(\boldsymbol{w_i}, \boldsymbol{r_i}) | \boldsymbol{i} = \boldsymbol{1}, \dots \boldsymbol{m_1}\}$,
$\boldsymbol{I_2} = \{(\boldsymbol{w_j}, \boldsymbol{r_j}) | \boldsymbol{j} = \boldsymbol{1}, \dots, \boldsymbol{m_2}\}$

Output: values of $\boldsymbol{K_n}(\boldsymbol{I_1}, \boldsymbol{I_2})$ for all $\boldsymbol{n} = \boldsymbol{1}, \dots, \infty$

Algorithm:

```
1. Pair all same words of the two images.
2. For each pair, calculate the relative location difference
   between them, and project the points to the offset space
   at location $\Delta \boldsymbol{r}$.
3. Cluster the pairs at the same location in the offset space
   to create a set of clusters S.
4. For each cluster in $\boldsymbol{S}$, we update the kernel values as
   follows: $\boldsymbol{K_i} = \boldsymbol{K_i} +_{|S|} \boldsymbol{C_i}$, where $|\boldsymbol{S}|$ is the size of this
   cluster.
```

2.4. Quantization

Till now, we have been assuming that the same feature has strictly the same layout. In practice, we need to loosen this constraint. In this section, we talk about how to code the geometry constraint of the special high order features.

It is straightforward to encode spatial information to 2nd order features. [11][18] use a spatial histogram centered at one of the word, and try to model different spatial distribution of the other word relative to this word. In [18], the histogram is only defined with distance, and [11] also separates the histogram in order to describe ‘above’, ‘below’, ‘to the left’ and ‘to the right’. It would be not easy to define the spatial histogram this way when it comes to 3rd or larger order features，since it is unclear how to define the relative position of one word to the other two words. We define the spatial layout for the features as follows. First we quantize the image space with a step size d as in Figure 2. The same visual words that are distributed in the same relative spatial layout in the quantized image are considered as the same feature. Therefore, the three words connected with the filled lines and those with the dotted lines for the two images in Figure 2 are considered as the same feature. For n distinct visual words, we can calculate the number of possible relative spatial layouts by fixing one word first and choose the possible location for the other words. Let w be the quantized width of an image, h be the quantized length. The number of possible relative spatial layout is as follows.

$$g_n = (2w-1)^{n-1}(2h-1)^{n-1}. \quad (1)$$

There would be fewer possible layouts when the n words in the feature are not distinct.

For the algorithm, what we only need to do is to replace the location r for each word with the quantized r. Therefore the offset of two pairs will also be the quantized offset.

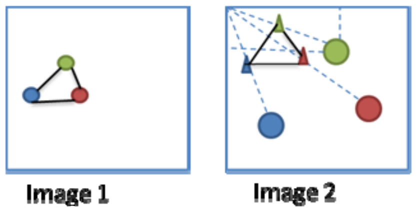


Figure 3: Method to handle scale invariant spatial features. Different colors represent different visual words. The size of the circle represents the scale of the word. The scales of the words in image 2 are twice the scales of the words in image 1.

2.5. Invariance

Till now, we have only considered translation invariance of our spatial high order features when calculating the kernels. We can also make the features (quasi-)scale invariant by making use of the scales of regions for the words [11]. To handle the scale invariance, we only need to make a small modification to our algorithm for the kernel calculation. Now an image I is represented as a collection of the word-location-scale triples (w_i, r_i, s_i).

As described in Figure 3, when we rescale the location r by the ratio of the scales of the words in the two images, a co-occurrence of the scale invariant feature becomes a constant shift of the same words from one image to the other again. Then, we can use the same algorithm to calculate the kernel functions. Therefore, we use the same algorithm as described in section 2.3, except at step 2, we calculate Δr as $r_1 - r_2 \times s_1/s_2$, rather than $r_1 - r_2$.

In similar way, it is also possible to make the features rotation invariant by using the dominant orientation of the region.

2.6. Computational Time

Let M represent the number of features in an image. It

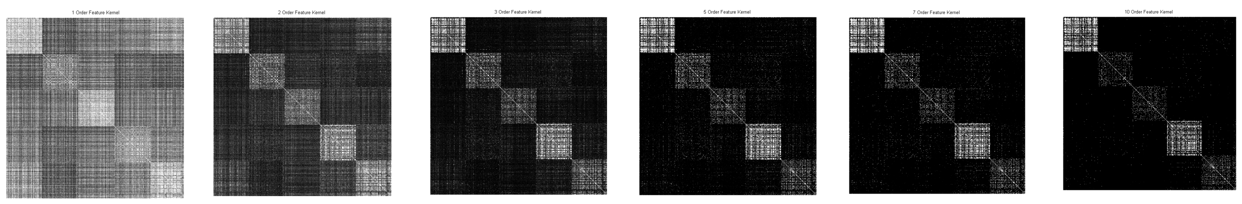


Figure 4: Kernel Matrix of the training data for different Order features. From the left to right are 1st, 2nd, 3rd, 5th, 7th, 10th order kernel matrices. Training data is arranged in the categories of 'face', 'airplane', 'rear car', 'motorbike', and 'watches'. As the order increases, the matrix gets sparser and sparser.

takes $O(P)$ time to pair the same words in two images and calculate the offsets of the pairs (step 1 and 2), where P is the number of pairs created. To cluster the pairs with the same offsets and update the kernel needs $O(P)$ time. In the worst case (only one visual word occurs in each image), we have $O(M^2)$ pairs. However, in practice we can only form pairs linear to M. Especially with a large codebook size, the number of pairs we form can be even smaller than M. Moreover, after running one time of the algorithm, we have already calculated all order kernels. Therefore, in practice we only need $O(M)$ time to calculate the kernels of all order features, which is the same complexity with calculating a kernel with bag of words model.

3. Experiments

We evaluate the proposed kernel for object categorization task. We use several public datasets: Caltech-101, MSRC dataset [18], and Graz-01 dataset. For all the datasets, we use the whole images (No bounding box) for both training and testing, which we believe is an advantage for our algorithm, since the high order kernels support partially matching similarities. Figure 9 gives some example co-occurring high order features found by the proposed algorithms.

3.1. Implementation Issues

For all the datasets, we apply Harris-Laplacian interest point detector [15] and SIFT detector [13] to each gray scale image to get the local features. The local features are further clustered with K means algorithm to obtain the visual words. We apply the proposed kernels to K nearest neighbor and Support Vector Machine for the classification task. For the implementation of SVM, we used the public library Libsvm [2]. For stableness, we do not consider scaling and rotation in these experiments.

Due to the large dimension of the feature space, images will have extremely sparse representations in the feature space. This also leads to the fact that the kernel based self-similarity of an image will be much larger than the similarity between two distinct images. Thus, our kernel matrix will be nearly diagonal, especially for large order. This is called *diagonal dominance* in machine learning, and is proved to be a problem when the kernel matrix is applied to learning algorithms such as SVM. Many methods have been proposed to overcome this problem [6]. We applied the negative diagonal shift method, which is to subtract a constant from the diagonal of the kernel matrix. Although it is possible make the kernel matrix not positive semi-definite any more, it has shown to gain good performance in practice.

3.2. Effect of High Order Features

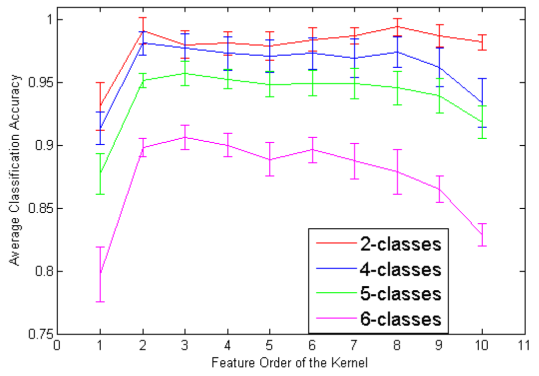
We use six object categories from the Caltech 101 dataset: $\{$faces, motorbikes, airplanes, rear cars, watches, ketches$\}$. This dataset has previously been used for unsupervised object category detection [7], while we use it in a supervised way. The goal of this experiment is to explore the change of the classification performance and computational time as we increase the order of the features. For all the six categories, there are more than 100 images per category, and reliable interest point can be detected. The task is to classify an image to one of the six categories. For the experiments, we randomly choose 50 images per category for training and 50 other images for testing. We repeat this 10 times and present the average results. We set the dictionary size as 500. We report the results of classification for two classes (faces and motorbikes), four classes (faces, motorbikes, airplanes, and rear cars), five classes (4 classes + watches), and six classes. We first use the step size 16 for image quantization in the experiments. For SVM, we use the one-vs-one scheme implemented in Libsvm for the multi-class classification.

Kernel Matrix

Figure 4 shows the kernel matrix of the training data with five classes for different order features. As the order increases, the matrix gets sparser and sparser, thus more discriminative among different categories. However, the inner class similarity matrix also gets sparser for some categories, such as airplane and rear car, with large orders (7 or 10). This is due to the fact that few co-occurring large order features are detected for some image pairs in the same category when the category includes more variance in object structure, scale, or rotation. Faces and Motorbikes are two objects with the most consistent structures.

Individual vs. Cumulative

We show the classification results with K Nearest Neighbor classifier ($k = 3$) in Figure 5. We use the



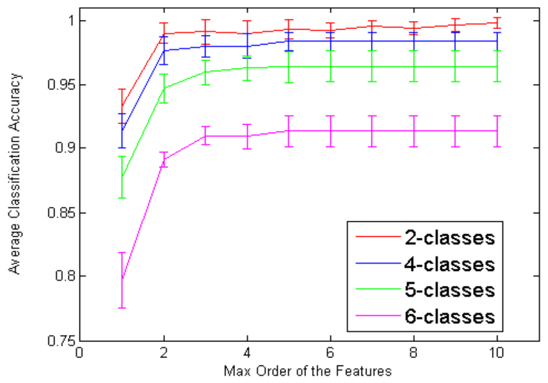


Figure 5: Object categorization performance with KNN (k=3). We show the average classification accuracy for classification task with 2, 4, 5, and 6 classes. The x axis is the feature order n. The left figure shows the accuracy only using the n^{th} order kernel. The right figure shows the accuracy using the weighted sum of the kernels from 1 to n.

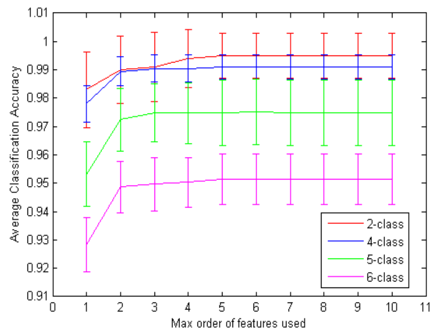


Figure 6: Object categorization performance with SVM. We use weighted sum of the individual kernels

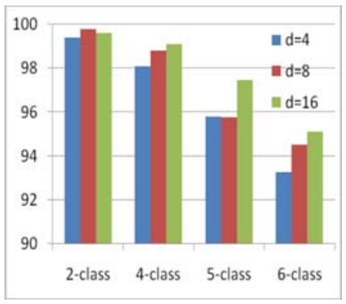


Figure 7: Average classification accuracies when using different step size for quantization.

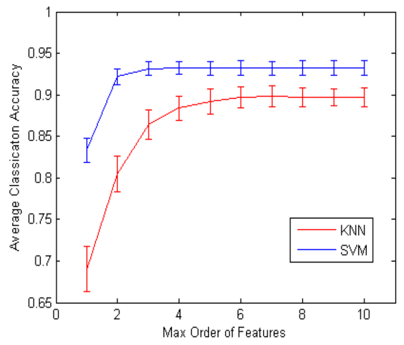


Figure 8: Categorization accuracies with dictionary size 50. We show the results for 6 class classification.

normalized kernel as the similarity function for KNN. We present both the results of individual kernels k_n with only n^{th} order features, and the results of weighted sum of kernels from 1st to n^{th} order individual kernels. We use $\mu = 0.02$ for the weighting. We show the performance until 10th order since most images do not have co-occurring features whose order are larger than 10. We found that for both individual and cumulative kernels, we gain significant improvement from 1st order (bag of words) to 2nd order features, which proves that modeling geometry information helps a lot. For the individual case, we get best performance with 8th order kernel for 2 class classification task, 2nd for 4 class, and 3rd for both 5 and 6 class classification task. The accuracy dropping for the individual kernels is mainly because as the order increases, fewer images will have co-occurring features as we have seen in the kernel matrix (Figure 4). For the cumulative case, the accuracy generally keeps increasing as we increase the order (may stop growing at some order). We found that even if individually the accuracy drops for the large order kernels, they may still contribute to the performance when we add them together. This is not surprising given the fact that higher order features are more discriminative. We reach best performance when the maximum order is 10 for 2 class, 5 for 4, 5, and 6 class classification tasks.

Using as Kernel for SVM

We apply the kernel to SVM. Figure 6 shows the results for the weighted sum of the individual kernels. Generally, the accuracy with SVM is better than KNN for all the orders. We can still see the performance increases as we increase the order. We reach the best performance when the max order of the features used is around 5.

Quantization Step Size

Figure 7 shows the average classification accuracy when we use different step sizes to quantize the image. A larger step size will give more geometry constraint to the features. The performance is similar when we have fewer classes. As the number of classes increases, a larger step size gives better performance, since we designed the experiments as categories with more variance are added generally.

	1 Order	1 - 2 order	1˜10 order
KNN	66.0±4.6	71.3±2.9	73.1±3.0
SVM	78.1±4.3	78.3±2.6	80.4±2.5

Table 1: Average classification accuracy on 9 categories of MSRC-2 dataset.

	Proposed	[9]	[17]
Bike	94%	84%	89.6%
Person	84%	82%	80.8%

Table 2: Equal Error Rate for categorization with Graz 01 dataset (no bounding box is used). We use the combined kernel for 1st to 10th order features.

Dictionary Size

The experiments till now are using a dictionary size 500. We try to decrease the dictionary size to 50. Figure 8 shows the classification performance when using KNN and SVM. In this case, the individual visual words would be quite meaningless, which results in low accuracies when using low order features. Especially for KNN, the accuracy is under 70% when using bag of words model. However, for both KNN and SVM, as we increase the order of the features, we got the performance quite close to that when we use 500 visual words. This shows that even if the local features are not discriminative itself, by increasing the orders, we can create discriminative features.

Computational Time

It takes averagely 0.02 second for computing the proposed kernels between two images with a Matlab implementation on a 3.4GHz CPU, when we have around 500 words per image. This is the time for computing kernels with all order features (1st, 2nd to any large order), since we only need to run the algorithm once to get those kernels.

3.3. Increasing the Variance

We further perform experiments with the MSRC-2 dataset [18], which has much more variance within the same category than the Caltech 101 dataset. There are 15 categories, and 30 images per category in this dataset. Previous work using this dataset for object categorization removes the background cluster for both training and testing. However, we evaluate on the whole image. We choose nine categories out of fifteen: {cow, airplanes, faces, cars, bikes, books, signs, sheep and chairs}, so that objects from different categories won't appear in the same image. This is a difficult dataset, since in [20], even if with a bounding box, rather than removing the background cluster completely, the accuracy drops to 78.5%.

Following the standard set up, we randomly select 15 images per category for training, and the rest for testing. We repeat this ten times and report the average performance. Table 1 shows the experiment results with both SVM and KNN. The results show that high order features improve the classification accuracy.

We also applied our algorithm to the Graz-01 datasets. We use the same dataset as in [17], which contains 100 training images and 50 testing images for each category. The task is to classify an object category (person or bike) from the background category. We show the Equal Error Rates in table 2. We compare our results with [9] and [17].

True Label	Inferred label								
	cow	airplane	face	car	bike	book	sign	sheep	chair
cow	12	0	0	0	0	0	0	1	2
airplane	0	14	0	1	0	0	0	0	0
face	0	0	12	2	0	0	0	0	1
car	0	0	1	12	0	0	0	0	2
bike	0	0	0	0	15	0	0	0	0
book	0	0	1	0	0	12	0	0	2
sign	1	0	2	0	1	1	9	0	1
sheep	1	0	0	0	0	0	0	13	1
chair	0	0	1	0	1	0	1	0	12

Table 3. Confusion Matrix of MSRC dataset. There are 15 images per category. This is the result of one run among the ten runs of the experiment. (Accuracy = 82.2%)

In [17], they mainly focus on local appearance, while [9] ignore local appearance, and but use pairwise interactions between simple features to model the shape of the objects.

4. Conclusion

We proposed an efficient algorithm to calculate kernels with high order spatial features. The algorithm is based on a transformation from the image space to the offset space, which makes the task of finding any order co-occurring features quite easy and fast. The computational complexity of the kernel is proved to be linear to the number of features per image, and does not depend on the order of the features. This enables us to use any large order features. The experiment results of object categorization task showed that high order features are useful, and the performance will usually increase as we increase the feature order. The improvement is especially obvious when the local features are not discriminative enough

We believe that there are still a lot of improvement we can do for this approach. First, although we showed the possible solution for scale and rotation invariance, we did not use them in the experiments. It still needs to be explored for the robustness when we try to deal with scaling and rotation with the algorithm. Second, currently we are doing quantization on the image space in order to loosen the geometry constraint. It can be beneficial to do the quantization directly on the offset space. Finally, although we showed that the kernel has the same computational complexity with bag of words model, it would be interesting to explore how to use it efficiently in a large dataset, where bag of words can take the advantage of their vector representation.

Acknowledgement

The work is supported in part by Industrial Technology Research Institute and Chunghwa Telecomm.

References

[1] A. Bosch, A. Zisserman, and X. Munoz. Representing shape with a spatial pyramid kernel. In CIVR, 2007.

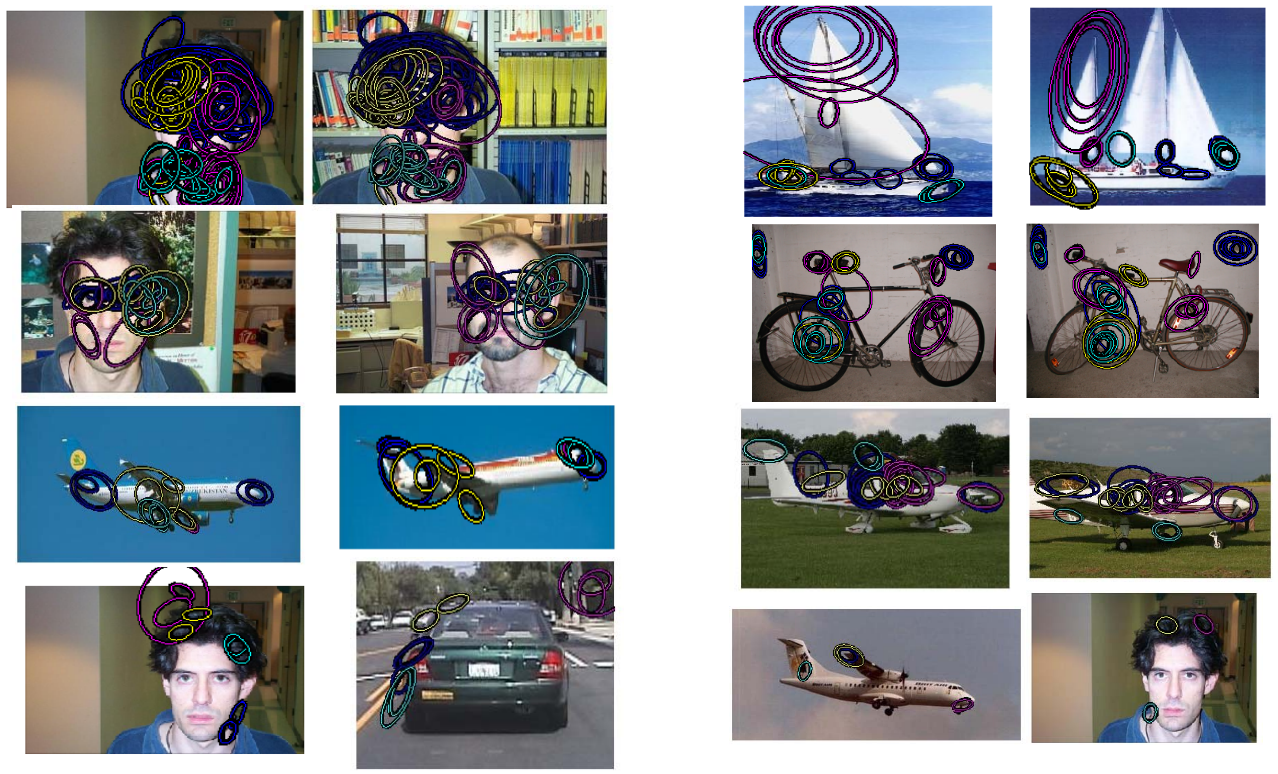


Figure 9: Example High Order features found for two images. For each pair of images, we show the features of the largest 4 orders between them. Different color represents different co-occurring features. For the first three rows, the two images are from the same category. For the last row, images are from different categories. The results show that we can find larger order features for images from the same category.

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

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

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

[5] K. Grauman and T. Darrell. The pyramid match kernel: discriminative classification with sets of image features. In *ICCV*, 2005.

[6] D. Greene and P. Cunningham. Practical solutions to the problem of diagonal dominance in kernel document clustering. In *ICML*, 2006.

[7] G. Kim, C. Faloutsos, and M. Hebert. Unsupervised modeling and recognition of object categories with combination of visual contents and geometric similarity links. In *ACM International* Conference on Multimedia Information Retrieval *(ACM MIR)*, October 2008.

[8] M. Leordeanu and M. Hebert. A spectral technique for correspondence problems using pairwise constraints. In *ICCV*, 2005.

[9] M. Leordeanu, M. Hebert, and R. Sukthankar. Beyond local appearance: Category recognition from pairwise interactions of simple features. In *CVPR*, 2007.

[10] H. Ling and S. Soatto. Proximity distribution kernels for geometric context in category recognition. In *ICCV,* 2007.

[11] D. Liu, G. Hua, P. Viola, and T. Chen. Integrated feature selection and higher-order spatial feature extraction for object categorization. In *CVPR*, 2008.

[12] H. Lodhi, C. Saunders, N. Cristianini, C. Watkins, and B. Scholkopf. Text classification using string kernels. *Journal* of Machine Learning Research, 2:563–569, 2002.

[13] D. G. Lowe. Distinctive image features from scale-invariant keypoints. International Journal of Computer Vision, 60:91–110, 2004.

[14] K. Mikolajczyk, B. Leibe, and B. Schiele. Multiple object class detection with a generative model. In *CVPR*, 2006.

[15] K. Mikolajczyk and C. Schmid. *Int. J. Comput. Vision*, (1):63–86, October.

[16] J. Niebles and L. Fei-Fei. A hierarchical model of shape and appearance for human action classification. In *CVPR*, 2007.

[17] A. Opelt, A. Pinz, M. Fussenegger, and P. Auer. Generic object recognition with boosting. In *PALM*, 2006.

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

[19] E. B. Sudderth, A. Torralba, W. T. Freeman, and A. S. Willsky. Learning hierarchical models of scenes, objects, and parts. In *ICCV*, 2005.

[20] J. Winn, A. Criminisi, and T. Minka. Object categorization by learned universal visual dictionary. In *ICCV*, 2005.