Spatial Pooling of Heterogeneous Features for Image Applications

Lingxi Xie*
Tsinghua University
Beijing 100084, China
198808xc@gmail.com

Qi Tian
Univ. of Texas at San Antonio
Texas, TX 78249
qitian@cs.utsa.edu

Bo Zhang
Tsinghua University
Beijing 100084, China
dcszb@mail.tsinghua.edu.cn

ABSTRACT

The Bag-of-Features (BoF) model has played an important role for image representation in many multimedia applications. It has been extensively applied to many tasks including image classification, image retrieval, scene understanding, and so on. Despite the advantages of this model such as simplicity, efficiency and generality, there are also notable drawbacks for this model, including poor power of semantic expression of local descriptors, and lack of robust structures upon single visual words. To overcome these problems, various techniques have been proposed, such as multiple descriptors, spatial context modeling and interest region detection. Though they have been proven to improve the BoF model to some extent, there still lacks a coherent scheme to integrate each individual module.

To address the problems above, we propose a novel framework with spatial pooling of heterogeneous features. Our framework differs from the traditional Bag-of-Features model on three aspects. First, we propose a new scheme for combining texture and edge based local features together at the descriptor extraction level. Next, we build geometric visual phrases to model spatial context upon heterogeneous features for mid-level representation of images. Finally, based on a smoothed edgemap, a simple and effective spatial weighting scheme is performed on our mid-level image representation. We test our integrated framework on several benchmark datasets for image classification and retrieval applications. The extensive results show the superior performance of our algorithm over state-of-the-art methods.

Categories and Subject Descriptors

H.3.1 [**Information Storage and Retrieval**]: Content Analysis and Indexing

*Lingxi Xie and Bo Zhang are from the State Key Laboratory of Intelligent Technology and Systems, Tsinghua National Laboratory for Information Science and Technology (TNList), and Department of Computer Science and Technology, Tsinghua University.

MM'12, October 29–November 2, 2012, Nara, Japan.

General Terms

Algorithms, Experimentation

Keywords

BoF Framework, Image Representation, Complementary Descriptors, Geometric Visual Phrases, Phrase Pooling, Spatial Weighting, Multimedia Applications

1. INTRODUCTION

Today, there are many multimedia applications based on image understanding and processing, such as image retrieval, image classification, scene understanding, and so on. Although different tasks vary in their goals, they share a key idea seeking for discriminative representations of images.

Traditional Bag-of-Features (BoF) framework has been widely used for various image applications mentioned above. It aims at providing better representations for images. For this purpose, local descriptors such as SIFT [16] are extracted from images, and a codebook is built upon all descriptors, depressing noises and forming a semantic visual vocabulary for the dataset. Finally, descriptors are quantized onto the codebook, and visual words are pooled for a statistical representation of the original image.

Despite the great success of BoF framework, there still exist many drawbacks in it. These drawbacks come mainly from the well-known semantic gap [17] between low-level local descriptors and high-level image semantics. Many researchers such as Yuan *et.al* [24] have noticed that SIFT descriptor suffers from both synonymy and polysemy. To overcome, the following schemes are widely adopted.

- **Different descriptions of local patches.** For single descriptors might fail to capture the rich information within local patches, it is reasonable to extract multiple descriptors for compensation. Various descriptors represent local patches from different aspects, providing more descriptive and discriminative information. By simply concatenating appearance and shape features together, [2] outperforms state-of-the-art models using single descriptors by a margin.
- **Mid-level representation connecting low-level and high-level concepts.** In BoF framework, an image is represented as a large set of visual words. However, there is a big semantic gap between low-level features and high-level concepts [17]. Therefore, many researchers have proposed some mid-level representations, such as macro-descriptors [3] or visual

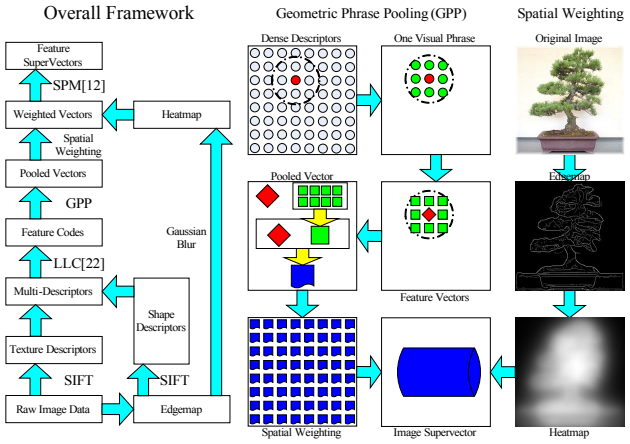

Figure 1: Flowcharts and of the framework and key modules. Left: the modified BoF framework. Right: Geometric Phrase Pooling and spatial weighting.

phrases [24], for better image understanding. Both of them bridge the semantic gap to some extent.

- **Spatial weighting of images.** Not all regions on a natural image are really useful for representation. Background clutters might bring in noises, which are harmful to training robust models. Therefore, detection of Regions of Interest (ROI) is usually proposed. Using a simple rectangular bounding box, [2] gives better representation on the less aligned datasets.

Based on these observations, we propose several novel algorithms from new aspects. With the help of edge detectors, we obtain an edgemap for each original image. Upon this, we claim a threefold contribution. First, we simultaneously extract SIFT and Edge-SIFT descriptors and combine them in the generation of BoF model. Earlier fusion of descriptors makes it easier to mine complementary information from both descriptors. Second, we propose geometric visual phrases upon traditional visual words, and take them as mid-level image representation as well as apply a novel pooling algorithm to them. Third, we use naive Gaussian blur to obtain a weighting heatmap for spatial weighting on the image plane. Integrating all the techniques produces a much more powerful framework, which outperforms state-of-the-art systems by a margin on various applications.

The proposed algorithm is illustrated in Figure 1. Compared with the traditional BoF framework, some differences should be marked. First, we generate edgemaps from original images using the Compass Operator [19], an improved version of Canny Operator [4], laying an important foundation of our algorithm. On original images as well as edgemaps, we extract two kinds of descriptors, *i.e.*, SIFT and Edge-SIFT, and mix them as a large set of local features. After codebook is constructed and descriptors are quantized onto the visual vocabulary, we build geometric visual phrases as the mid-level representation of our model. Before the traditional max-pooling step, we insert two new steps, *i.e.*, phrase pooling and spatial weighting, for better description of geometric visual phrases. Finally, we apply SPM [12] for spatial context modeling, and obtain the representation vectors of images.

The remainder of this paper is organized as follows. In Section 2, we introduce the traditional Bag-of-Features framework for image applications. Section 3 presents our algorithm extracting and fusing complementary descriptors, as well as provides a detailed analysis of the strategy. In Section 4, we introduce a new pooling algorithm named Geometric Phrase Pooling. It is very effective to find structural information beyond visual words. Section 5 introduces a simple spatial weighting scheme for interesting region detection. Various image applications of our framework are shown in Section 6. We draw our conclusions in Section 7.

2. THE BAG-OF-FEATURES FRAMEWORK

The Bag-of-Features (BoF) framework is one of the most widely used models for image representation. In this section, we give a brief introduction, and build a mathematical notation system for this framework.

2.1 Local Descriptor Extraction

In computer, an original **image I** is a $W \times H$ matrix:

$$\mathbf{I} = (a_{ij})_{W \times H} \tag{1}$$

where a_{ij} is the **pixel** on position (i, j). For grayscale images, a_{ij} is a single floating number ranged in $[0, 1]$, while it is a 3-dimensional vector for RGB-space color images.

Due to the poor semantic meaning of raw image pixels, we extract **local descriptors** from small patches on the image plane. There are many works on describing local patches. Among those, SIFT [16] and HOG [6] are probably the most widely used ones. They are both gradient based descriptors extracted on **Interest Points** of images. Detecting Interest Points is also a challenging problem. Since many detectors such as DoG [16] or MSER [18] sometimes fail to find semantic and discriminative patches. we use an alternative method by performing dense sampling of local patches, leading to the **Dense-SIFT** or **Dense-HOG** algorithm [1].

After descriptor extraction, the image $\mathbf{I}$ could be represented as a set of local descriptors, $\mathcal{M}$:

$$\mathcal{M} = \{(\mathbf{d}_1, \mathbf{l}_1), (\mathbf{d}_2, \mathbf{l}_2), \ldots, (\mathbf{d}_M, \mathbf{l}_M)\} \tag{2}$$

where $\mathbf{d}_m$ and $\mathbf{l}_m$ denote the D-dimensional description vector and the geometric location of the m-th descriptor, respectively. M is the total number of dense descriptors, which could be hundreds or even thousands under dense sampling.

Note that SIFT or HOG descriptors only provide texture features of images. By extracting various kinds of descriptors, it is possible to cover other important features such as shape, color, and so on. Many systems on this topic, such as [2], have been proposed, showing a much better performance over those using single descriptors.

2.2 Quantization for Descriptors

After descriptors have been extracted, they are often quantized to be compact. For this purpose, we train a **codebook B** using descriptors from the whole dataset. It is a $B \times D$ matrix, or B vectors with dimension D, each of which, *i.e.*, $\mathbf{c}_b$, is called a **codeword**. Most often, the codebook is constructed with K-Means clustering algorithm. Recent years, there are also some works targeting at improving the performance of K-Means [14] [20], and building discriminative codebooks for large-scale image applications [26].

Next, descriptors are projected onto the codebook for a histogram representation. This process is called **coding**, for

we are actually encoding each descriptor into a sparse vector. Hard quantization strategy presents a descriptor using single codeword, leading to a large quantization error. In recent years, soft quantization methods have been proposed as alternatives to the hard one. By projecting a descriptor onto the subspace spanned by a small group of codewords, it produces smaller quantization error, and performs more effective. Sparse Coding [13] [23] and Locality-constrained Linear Coding [22] techniques are such cases. Given a codebook with B codewords, the quantization vector or **feature vector** for a descriptor $\mathbf{d}_m$ would be a B-dimensional vector $\mathbf{v}_m$. We name $\mathbf{v}_m$ the corresponding **visual word** of **descriptor** $\mathbf{d}_m$. In our terminology, there is a clear difference between the concepts of descriptors and visual words.

2.3 Feature Pooling

After all the descriptors are quantized as visual words, we shall aggregate them for image representation. We call this step **feature pooling**, for we are putting visual words into a pool for statistics. For this purpose, two major pooling strategies are often used. The **max-pooling** strategy calculates the maximal responses on each codeword:

$$\mathbf{w} = \max_{1 \le m \le M} \mathbf{v}_m \tag{3}$$

where the notation $\max_m$ denotes the element-wise maximization, M denotes the number of local patches of the image, and $\mathbf{v}_m$ is the m-th visual word. Differently, the **average-pooling** strategy calculates the average responses:

$$\mathbf{w} = \frac{1}{M} \sum_{m=1}^{M} \mathbf{w}_m \tag{4}$$

Here $\mathbf{w}$, a D-dimensional vector, is named **representation vector** or **feature vector** of the image.

Some researchers [3] have discussed the choice of max-pooling versus average-pooling. Max-pooling gives more discriminative representation under soft quantization strategies, while average-pooling fits hard quantization better. Recently, various methods have been proposed to improve the traditional pooling methods, such as the complex optimization in the Geometric ℓ_p-norm Pooling [9] algorithm.

2.4 Spatial Context Modeling

Spatial context could greatly help us understand the semantics of images [29]. Therefore, various models are used for constructing spatial structures. Among those, Spatial Pyramid Matching [12] is a successful trial. After coding, images are divided into hierarchical subregions for individual pooling, and the pooled vectors are concatenated to form a **supervector**. For datasets with good alignment such as Caltech101 [8], SPM improves classification accuracy by an impressive margin of 10%. However, it shows little improvements or even worse performances on less aligned datasets.

Another line for spatial context modeling is to use visual phrases [15] [27] [28]. Compared with visual words, visual phrases are more semantic and robust [25], therefore produce more discriminative vectors for image representation. Also, spatial coding of visual phrases are widely used in large-scale web image search and retrieval systems [29].

2.5 The Baseline Systems

We compare our model with two baselines. The first is LLC [22], a standard implementation of traditional BoF framework with single SIFT descriptors. The other one is the system by Bosch *et.al.* [2]. In [2], various techniques are exploited, including multiple descriptors, detection of regions of interest, and so on. Both baselines represent state-of-the-art performances for image representation.

3. COMPLEMENTARY DESCRIPTORS

In this section, we propose a novel system for image representation using two kinds of descriptors. First, we introduce an edge based descriptor named Edge-SIFT. Though they come from heterogeneous domains of original images and edgemaps, SIFT and Edge-SIFT descriptors share the same physical meaning in their corresponding dimension, therefore we could mix them to train a BoF model. We test our model and perform detailed analysis on the effect of multiple descriptors. Also, we make a short comment on the early fusion strategy to reveal its advantages.

3.1 SIFT and Edge-SIFT Descriptors

For an image $\mathbf{I}$, we first extract dense SIFT [16] descriptors from the image. Denote the set of SIFT descriptors as $\mathcal{M}_{\mathrm{S}}$:

$$\mathcal{M}_{\mathrm{S}} = \{(\mathbf{d}_{\mathrm{S}1}, \mathbf{l}_{\mathrm{S}1}), (\mathbf{d}_{\mathrm{S}2}, \mathbf{l}_{\mathrm{S}2}), \ldots, (\mathbf{d}_{\mathrm{S}M_{\mathrm{S}}}, \mathbf{l}_{\mathrm{S}M_{\mathrm{S}}})\} \tag{5}$$

where the subscript S stands for SIFT, and M_{S} is the number of SIFT patches on the image plane.

As we know, SIFT descriptors are effective on describing texture features. To compensate, we can also extract shape descriptors. Following [2], we apply an edge detector on image $\mathbf{I}$, producing another $W \times H$ grayscale image $\mathbf{I}_{\mathrm{E}}$:

$$\mathbf{I}_{\mathrm{E}} = \{e_{ij}\}_{W \times H} \tag{6}$$

where e_{ij}, a floating value ranged within $[0, 1]$, is the significance quantity for pixel (i, j) locating on an edge. We call $\mathbf{I}_{\mathrm{E}}$ the corresponding **edgemap** for original image $\mathbf{I}$.

We use Compass Operator [19] for edge detection. One of the detected edgemaps is shown in Figure 2, while more of them could be found in Figure 4 and 13. On edgemaps, texture details are filtered and the shape of the objects becomes more clear. Therefore, it is reasonable to extract SIFT descriptors on edgemaps for shape description. We call them Edge-SIFT descriptors, to differ from original SIFT ones. Denote the set of Edge-SIFT descriptors as $\mathcal{M}_{\mathrm{E}}$:

$$\mathcal{M}_{\mathrm{E}} = \{(\mathbf{d}_{\mathrm{E}1}, \mathbf{l}_{\mathrm{E}1}), (\mathbf{d}_{\mathrm{E}2}, \mathbf{l}_{\mathrm{E}2}), \ldots, (\mathbf{d}_{\mathrm{E}M_{\mathrm{E}}}, \mathbf{l}_{\mathrm{E}M_{\mathrm{E}}})\} \tag{7}$$

where the subscript E stands for Edge-SIFT, and M_{E} is the number of descriptors, which could be different with M_{S}.

3.2 Fusing Descriptors

After different descriptors are extracted, we simple mix them on the same image plane:

$$\mathcal{M} = \mathcal{M}_{\mathrm{S}} \cup \mathcal{M}_{\mathrm{E}} \tag{8}$$

Here, $\mathcal{M}$ is the new set of descriptors for image representation. Figure 2 illustrates the fusion operation.

We shall emphasize that both SIFT and Edge-SIFT descriptors are 128-dimensional histograms of gradients, therefore share the same physical meaning though extracted from different sources of images. Note that it is important to guarantee the same physical meaning on the corresponding dimensions, for we shall compare and combine the descriptors dimension-wise at the clustering and quantization steps.

Finally, we shall make a short comment to compare our work with [2]. In [2], descriptors are also extracted from

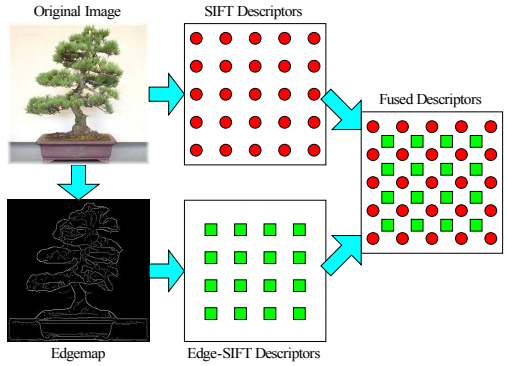

Figure 2: Fusing two categories of dense descriptors on the image plane.

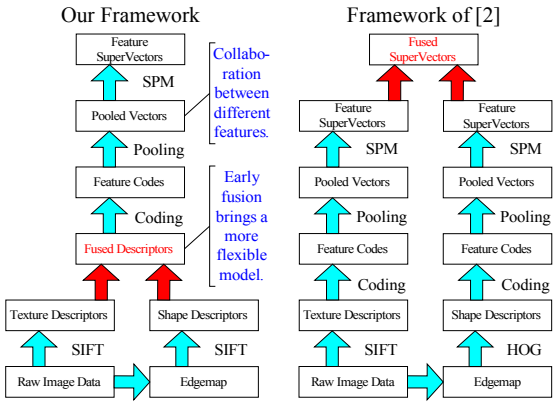

Figure 3: Difference between our model and [2] (best viewed in color PDF). Red arrows and texts highlight the fusion step in both frameworks.

original images and edgemaps respectively. However, two categories of descriptors are individually processed through BoF framework, until fusion is performed on the BoF representation of images to form a concatenated supervector. As we could see in Section 4.5, late fusion limits the flexibility of the model, and makes it impossible to construct spatial structures consisting of both kinds of descriptors. On the contrary, we finish the fusion step much earlier, obtaining a new set containing both categories of descriptors. We illustrate the difference between the models in Figure 3, and will further discuss the advantages of our model in Section 4.5.

3.3 Experiments and Discussions

Now, we test our model on the Caltech101 dataset. To make comparison, we inherit the remaining parts of our framework from LLC [22]. Table 1 shows our results on different combinations of descriptors as well as on single ones. When we fuse different kinds of descriptors, *i.e.*, SIFT and Edge-SIFT, our system gives the best performance, while we observe dramatic accuracy drops when the same kind of descriptors, *i.e.*, both SIFT, are merged. Therefore, the compensation between two kinds of descriptors is very clear.

To give a better intuition to this finding, we return to systems using single descriptors. We choose parameters from the best after-fusion performance in Table 1, *i.e.*, the step

Table 1: Results on combined descriptors as well as on single ones. The numbers in brackets are respectively step and patch sizes for SIFT extraction.

Desc 1	Desc 2	Only 1	Only 2	Both
SIFT(7,7)	SIFT(6,12)	74.41%	72.69%	75.14%
SIFT(7,7)	Edge(6,12)	74.41%	73.08%	78.75%
SIFT(7,7)	SIFT(7,12)	74.41%	73.77%	75.75%
SIFT(7,7)	Edge(7,12)	74.41%	72.89%	**78.94%**
SIFT(7,7)	SIFT(8,12)	74.41%	73.60%	75.32%
SIFT(7,7)	Edge(8,12)	74.41%	72.93%	78.92%

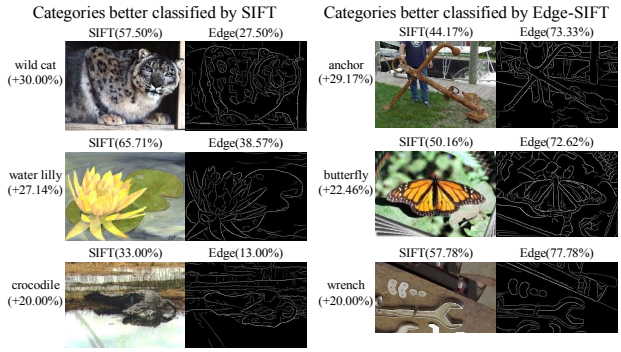

Figure 4: Comparison of different descriptors. Classification accuracies with single descriptors are listed above sample images.

sizes of SIFT and Edge-SIFT are both 7 pixels, while the sizes of local patches are chosen to be 7 and 12 pixels, respectively. To compare different descriptors, we separately evaluate the classification accuracies by category, and list those categories with largest accuracy gaps in Figure 4.

It is clear that different types of objects are better described using different types of descriptors. For objects with less deformation such as manmade tools, the better strategy is to filter their texture details and pay more attention on the edgemaps. Therefore, Edge-SIFT descriptors give a better description on these objects. However, there are also a number of objects in which texture features are more discriminative, while shape description is less robust. Animals and plants are such cases, where it is better to preserve texture details in original images and use SIFT descriptors.

Since different kinds of descriptors are complementary for discriminating objects, it is reasonable to preserve both of them for a more robust image representation. Figure 5 shows an example of four categories. It is hard to classify them using any single kind of descriptors, for the between-category similarity in texture or shape features. Only using both categories of descriptors makes it possible for our model to distinguish them well.

4. GEOMETRIC PHRASE POOLING

Till now, we have described a framework for image representation using two kinds of descriptors. However, it is worth noting that different descriptors are individually coded and pooled in the algorithm. The lack of cooperation between them limits the discriminativity of our model. In this

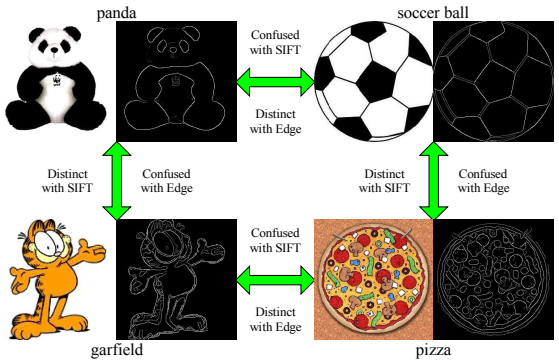

Figure 5: Combining different kinds of descriptors helps to discriminate confusing categories.

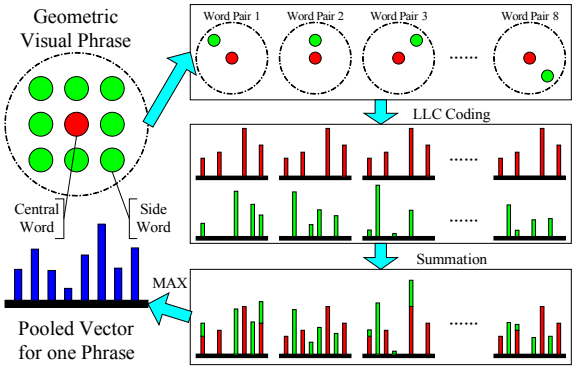

Figure 6: Illustration of Geometric Phrase Pooling. We group the central word and each side word as a pair and add their feature vectors together. At last, maximization is performed on the whole phrase.

section, we shall introduce a novel structure named geometric visual phrase. It shall provide us a mid-level representation containing both texture and shape features. We illustrate our algorithm in the right column of Figure 1.

4.1 The GPP Algorithm

Let's start from Equation (8), and rewrite it as:

$$\mathcal{M} = \{(\mathbf{d}_1, \mathbf{l}_1), (\mathbf{d}_2, \mathbf{l}_2), \ldots, (\mathbf{d}_M, \mathbf{l}_M)\} \tag{9}$$

To begin with, we give a new definition of geometric visual phrases. Following [24], we regard a visual phrase as a combination of visual words. By intuition, phrases often consist of words close to each other on the image plane. As local patches on images are not ordered so well as textual words, we ignore the order and take a phrase as a disordered set of visual words. Following this basic rule, we propose a simple algorithm. Define a positive integer K, $K \ll M$. For the m-th descriptor, we search for its K nearest neighbors on the image plane and form a word group:

$$\mathcal{G}_m = \{(\mathbf{d}_{m,0}, \mathbf{l}_{m,0}), \ldots, (\mathbf{d}_{m,K}, \mathbf{l}_{m,K})\}, \qquad m = 1, 2, \ldots, M \tag{10}$$

$\mathcal{G}_m$ is the m-th **geometric visual phrase**. The **central word** of $\mathcal{G}_m$ is defined as the zeroth descriptor $(\mathbf{d}_{m,0}, \mathbf{l}_{m,0})$, which is simply $(\mathbf{d}_m, \mathbf{l}_m)$ itself. The **location** of phrase $\mathcal{G}_m$ is defined by $\mathbf{l}_{m,0}$. Other K descriptors are called **side words**. K is the **order** of $\mathcal{G}_m$, which contains $K+1$ words.

Suppose that we have trained a codebook $\mathbf{B}$ with B codewords. For a phrase $\mathcal{G}_m$, we respectively perform LLC [22] coding for each of its words. LLC is a sparse coding scheme. Given $\mathbf{B}$ and number of bases r (most often $r \ll B$), it produces feature vectors with at most r nonzero elements among totally B dimensions. For $\mathcal{G}_m$, there will be $K+1$ sparse feature vectors, one for each visual word. Denote $\mathbf{v}_{m,k}$ as the feature vector of k-th word in $\mathcal{G}_m$.

Now, **Geometric Phrase Pooling** (GPP) is performed on the $K+1$ feature vectors:

$$\mathbf{w}_m = \max_{1 \le k \le K} \{\mathbf{v}_{m,0} + \mathbf{v}_{m,k}\} \tag{11}$$

$$= \mathbf{v}_{m,0} + \max_{1 \le k \le K} \mathbf{v}_{m,k} \tag{12}$$

where the notation $\max_k$ denotes element-wise maximization on K vectors with B dimensions.

Equation (11) is the core equation of GPP, while (12) is an equivalent version for easier implementation. We illustrate the simple working mechanism of (11) in Figure 6.

After GPP, we obtain a B-dimensional vector $\mathbf{w}_m$ as the feature vector for the m-th visual phrase $\mathcal{G}_m$. Finally, a max-pooling step is performed over all the phrases:

$$\mathbf{w} = \max_{1 \le m \le M} \mathbf{w}_m \tag{13}$$

and $\mathbf{w}$ is obtained as the feature vector of the image.

4.2 Deep Insights for GPP

The formulation of GPP is very easy to understand. However, the intuition behind the simple algorithm is not that straightforward. Here, we clarify the advantages within the formulation to give a better understanding of GPP.

First, we shall provide an intuitive explanation. Go back to Equations (11) and (13). Simple derivation gives:

$$\mathbf{w} = \max_{1 \le m \le M} \mathbf{w}_m \tag{14}$$

$$= \max_{1 \le m \le M} \left\{ \max_{1 \le k \le K} \{\mathbf{v}_{m,0} + \mathbf{v}_{m,k}\} \right\} \tag{15}$$

$$= \max_{1 \le m_1 < m_2 \le M, m_1 \diamond m_2} \{\mathbf{v}_{m_1} + \mathbf{v}_{m_2}\} \tag{16}$$

where $m_1 \diamond m_2$ means that visual words m_1 and m_2 are adjacent ones, *i.e.*, when one of them is taken as central word, the other is one of the side words. Hence, $\mathbf{w}$ is the maximization over summations of all adjacent word pairs. Define $\mathbf{v}_{m_1} + \mathbf{v}_{m_2}$ as the contribution to GPP from word pair (m_1, m_2).

Now, recall the formulation of max-pooling algorithm and rewrite it into an equivalent version:

$$\mathbf{w} = \max_{1 \le m \le M} \mathbf{v}_m \tag{17}$$

$$= \max_{1 \le m \le M} \left\{ \max_{1 \le k \le K} \{\max\{\mathbf{v}_{m,0}, \mathbf{v}_{m,k}\}\} \right\} \tag{18}$$

$$= \max_{1 \le m_1 < m_2 \le M, m_1 \diamond m_2} \{\max\{\mathbf{v}_{m_1}, \mathbf{v}_{m_2}\}\} \tag{19}$$

Naturally, $\max\{\mathbf{v}_{m_1}, \mathbf{v}_{m_2}\}$ is defined as the contribution to max-pooling from word pair (m_1, m_2).

For an adjacent word pair (m_1, m_2), consider its contributions to GPP and max-pooling. For simplicity, we denote feature vectors for them as $\mathbf{v}_1$ and $\mathbf{v}_2$, respectively. If $\mathbf{v}_1$

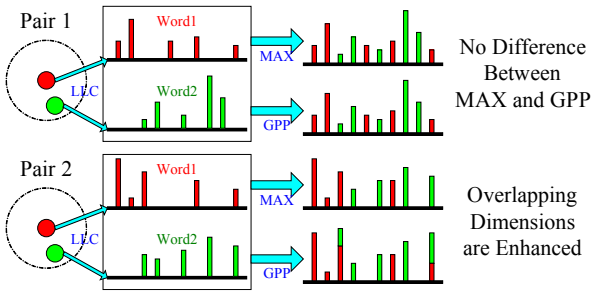

Figure 7: Intuition of GPP (best viewed in color PDF). Cases of non-overlapping word pair and partial overlapping word pair are shown respectively.

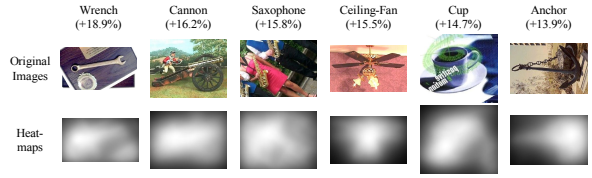

Figure 8: Heatmaps generated from the percentages of useful pairs of visual words on the image plane. Numbers in brackets are accuracy gains for those categories, from LLC to GPP.

and $\mathbf{v}_2$ have no nonzero dimensions in common, we have $\max\{\mathbf{v}_1, \mathbf{v}_2\} = \mathbf{v}_1 + \mathbf{v}_2$. In this case, the word pair contributes equally to max-pooling and GPP. However, if there are common nonzero dimensions in $\mathbf{v}_1$ and $\mathbf{v}_2$, things will be different: this word pair would better contribute to GPP by enhancing the overlapping dimensions. We illustrate our intuition in Figure 7.

From the analysis above, we have learned that only word pairs with nonzero dimensions in common would make contribution to GPP. In our framework, we perform descriptor quantization using LLC [22], a locality sensitive coding scheme. Descriptors are projected onto overlapping dimensions only if they are similar in feature space. Therefore, GPP latently selects those word pairs both similar and adjacent, and enhance their nonzero responses in common.

There is an acknowledged observation that on natural images, local patches with the same semantics are more likely to be correlated, *i.e.*, visually similar. Therefore for a geometric visual phrase consisting of one central word and K side words, the correlation between central word and side words will be high (visually similar) if this geometric visual phrase is located on a semantic patch. This high correlation turns to be the common nonzero dimensions of the visual words. Therefore, GPP enhances the responses of such highly correlated patches, or word pairs, in a latent way.

To verify our inference, we need some additional statistics. For each visual phrase, we count the percentage of side words with common nonzero dimensions with the central word. This could be taken as a histogram on the image plane. Smoothing it gives us a heatmap illustrating the visual correlations on the corresponding patches. Figure 8 shows some randomly selected examples from the Caltech101 dataset. We could see that significantly enhanced

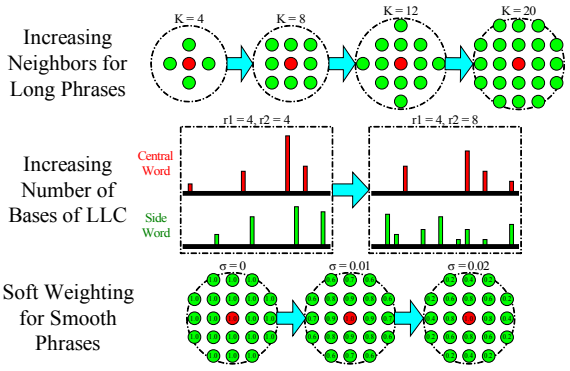

Figure 9: Illustration of different improvements for GPP (best viewed in color PDF). Above: increasing K for long phrases. Middle: Enlarging number of bases in LLC for more overlapping. Below: Soft weighting on geometric visual phrases, where floating numbers are the soft weights on visual words.

visual responses are mainly distributed on the semantic regions. By paying more attention on these regions, GPP produces better representations than LLC.

In summary, we could conclude that GPP is indeed a superior pooling method with a clear motivation and convincing improvements.

4.3 Enhancing GPP

In this section, we will discuss three ideas to improve the effectiveness of GPP. All of them are illustrated in Figure 9.

4.3.1 *Increasing Neighbors for Long Phrases*

In Section 4.2, we observe that word pairs with similar appearance and geometric location would contribute to GPP. To search for more such pairs, it is straightforward to increase the order of phrases, *i.e.*, K. However, increasing K would also bring in more irrelevant side words. We don't set K automatically, but choose the best performance ($K = 20$) after testing a wide range of K (from 5 to 40).

4.3.2 *Numbers of Bases of Visual Phrases*

As LLC [22] is a sparse coding algorithm, the number of bases r is much smaller than the visual codebook size B. In this case, histogram representations of adjacent visual words could hardly overlap, therefore percentage of word pairs that contribute to GPP is small. Increasing r produces more useful word pairs, but also damages the locality of LLC. The selection of r is a tradeoff between description and robustness.

The central word is the most significant component in a visual phrase, therefore its robustness is more important than side ones'. For this, we use a larger r for side words and yet a small r for central one. To clarify, we denote r_1 and r_2 as numbers of bases for central words and side words respectively. In our experiments, $r_1 = 5$ and $r_2 = 30$.

4.3.3 *Soft Weighting for Smooth Phrases*

By intuition, if the distance between visual words is large, the relationship between them is loose. Therefore, we apply an exponential decay on side words, assigning lower weights onto distant ones. For a side word $(\mathbf{d}_{m,k}, \mathbf{l}_{m,k})$ with feature

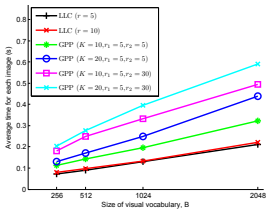

Figure 10: Average time required for each image. Data are tested on 4×2.0GHz CPUs.

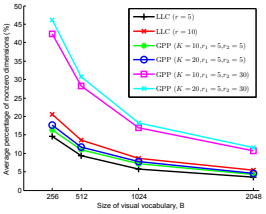

Figure 11: Average percentage of nonzero dimensions in pooled vectors.

vector $\mathbf{v}_{m,k}$, we penalize it with weight s_k defined as:

$$s_k = \exp\{-\sigma_w \times \|\mathbf{l}_0 - \mathbf{l}_k\|_2\} \tag{20}$$

where σ_w is the smoothing parameter on visual words, and $\|\cdot\|_2$ is the Euclidean distance. Now, we obtain a smoother version of GPP formulation (12):

$$\mathbf{w}_m = \mathbf{v}_{m,0} + \max_{1 \le k \le K} s_k \times \mathbf{v}_{m,k} \tag{21}$$

In all the experiments, we take $\sigma_w = 0.01$.

4.4 Time Complexity and Sparsity

Here, we test the time complexity and the sparsity of GPP to show its simplicity and efficiency. To make comparison, we use the implementation of LLC with code downloaded from the website provided by [22]. GPP with different parameters is added into the BoF framework as an intermediate step. We denote the original framework as LLC and our framework as GPP, construct different sizes of codebooks, and respectively record the average coding and pooling time for each image, as well as the average percentage of nonzero dimensions in the feature vector $\mathbf{w}$.

Results are plotted in Figure 10 and 11. Owe to the simplicity, our framework is very efficient to carry out. Geometric Phrase Pooling requires no more than 0.4s on a single image, which is much better than those complicated pooling algorithms such as GLP [9]. On the other hand, Figure 11 reveals that GPP generates much denser feature vectors for image representation, especially in the scenarios with longer phrases and more coding bases. A comment needs to be made here. People have been debating the use of dense versus sparse features for a long time. Yang *et. al* [23] have claimed the effectiveness of sparse feature vectors. Our experiments give a neutral argument about this topic: denser features do not mean to provide worse results, but we need proper structures to organize them. GPP actually provides a compromise between sparse and dense representations of images.

4.5 Early Fusion vs. Late Fusion

Finally, we return to the discussion of early fusion versus late fusion in Section 3. On the Caltech101 dataset, we test different combinations of fusion strategies and pooling methods. Results are listed in Table 2. From the table, we could clearly observe the advantage of early fusion over late fusion. When the geometric visual phrases are not built, they show comparable performances. However, GPP helps to construct robust mid-level structures upon fused descriptors, resulting in a notable improvement on the early fusion system. In

Table 2: Caltech101 classification results on frameworks with different fusion and pooling strategies.

	Late Fusion [2]	Early Fusion (Ours)
LLC [22]	79.12%	78.94%
GPP (Ours)	79.48%	81.36%

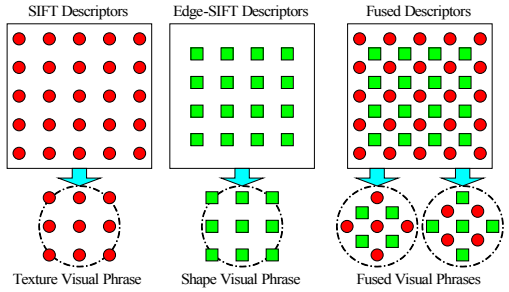

Figure 12: Applying GPP on fused descriptors helps to generate complementary visual phrases.

Figure 12, we illustrate the architectures of visual phrases over single and complementary descriptors. Containing both texture and shape features, mid-level structures upon fused descriptors are more robust and semantic meaningful.

5. SPATIAL WEIGHTING

For less aligned datasets, we need to detect Regions of Interest (ROI) on the original images. This is equivalent to a spatial weighting, or heatmap learning process. In this section, we provide a simple spatial weighting strategy using a gradient based edge detector, for the observation that higher contrast regions provide stronger stimuli to our vision [11].

Following (6), an edgemap $\mathbf{I}_\mathrm{E}$ is a $W \times H$ matrix with elements representing the intensities of edge responses. Based on it, we calculate a $W \times H$ **weighting matrix** $\mathbf{W}$:

$$\mathbf{W} = \{w_{ij}\}_{W \times H} \tag{22}$$

Here, w_{ij} is the spatial weight at position (i, j), which is accumulated from decayed edge responses:

$$w_{ij} = \sum_{i',j'} e_{i'j'} \times \exp\{-\sigma_e \left\|(i,j) - (i',j')\right\|_2\} \tag{23}$$

Here, coordinate (i', j') are traversed on the whole image, σ_e is the smoothing parameter on edge responses, and $\|\cdot\|_2$ is the Euclidean distance. As σ_e goes up, there shall be smaller weights accumulated on pixels far from edges. In our experiments, σ_e is fixed to 0.05.

With spatial weights, formulation (13) becomes:

$$\mathbf{w} = \max_{1 \le m \le M} \{w_m \times \mathbf{w}_m\} \tag{24}$$

where w_m is the weight at $\mathbf{l}_m$, the central pixel of $\mathcal{G}_m$.

To evaluate the proposed spatial weighting scheme, we test it on the Caltech101 dataset. Again we calculate the classification accuracies by category and list the most improved ones in Figure 13. For better visualization, we plot the corresponding weighting matrices $\mathbf{W}$ as heatmaps for

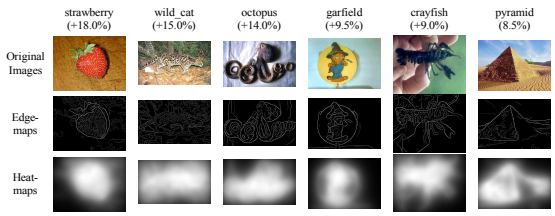

Figure 13: Edgemaps and weighting heatmaps of images from those categories with most accuracy gains (numbers in brackets) after spatial weighting.

comparison. Since higher weights often appear in the foreground, the feature vector obtained with (24) is more robust.

Finally let us note that (23) is very computing expensive, for it requires a complete enumeration on each pairs of pixels. On a 300×300 image, it might take more than 30 seconds on a single-core CPU. Here, we exploit an approximation by skipping the calculation of the pairs with Euclidean distances larger than 50 pixels. Taking our best smoothing parameter, *i.e.*, $\sigma_e = 0.05$, the maximal ignored coefficient could be $\exp\{-0.05 \times 50\} \approx 0.08$, which is relatively small. The approximation algorithm only requires less than 0.5 second on a single image. It is efficient in practise considering the calculation is required only once.

6. CASE STUDIES

In this section, we provide the experimental results on several widely used datasets with various tasks, *i.e.*, image classification, retrieval and scene understanding. To show the efficiency of our algorithm, we use the same settings as state-of-the-art algorithms, and the best settings learned from previous sections for GPP and spatial weighting.

- **Edge detection.** We use the Compass Operator [19] for edge detection. The radius parameter σ is fixed as 4 as proposed in the same literature.
- **Image descriptors.** We use the VLFeat [21] library to extract SIFT descriptors. The step size and the scale of patches are discussed separately for each dataset.
- **Codebook construction.** We use a traditional K-Means algorithm for clustering. The size of codebook is 2048 for Caltech101 and 15-scene dataset and 4096 for other two. The number of descriptors collected for clustering does not exceed 2 million.
- **Coding and pooling method.** We apply the original source code provided by authors of LLC [22] for coding. Upon that, we apply GPP with the best parameters: $K = 20$, $\sigma_w = 0.01$, $r_1 = 5$ and $r_2 = 30$.
- **Spatial weighting.** We take $\sigma_e = 0.05$ for edge based spatial weighting scheme.
- **SPM settings and normalization.** We apply a 3-layer SPM for enhancing the spatial context of features. After that, a ℓ_2-norm normalization is performed to produce comparable feature vectors.
- **SVM for tagging.** A linear SVM by Chang *et.al* [5] is used for training and testing. For retrieval tasks, we rank the images according to the confidence probabilities reported by SVM.
- **Accuracy evaluation.** For Pascal VOC 2007 Challenge, we use the standard benchmark method. On other datasets, we randomly selected fixed number of images for training, and test the model on others to evaluate the average classification accuracy over all categories. We repeat random selection for 10 times and report the average performance.

Figure 14: Sample images from Pascal VOC 2007 dataset. Upper: images with objects of multiple categories. Middle: images with small objects. Bottom: images with many objects of same category.

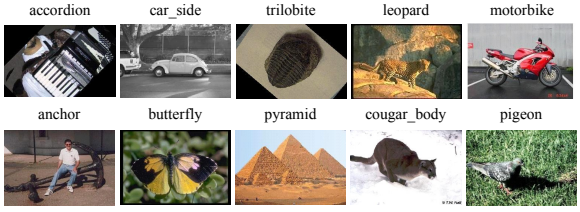

Figure 15: Sample images from Caltech101 dataset. Upper: categories on which our framework reports 100% accuracies (30 trains). Bottom: categories on which our framework most outperforms LLC [22].

6.1 Pascal VOC 2007 Dataset

There are 9963 images containing 20 categories of objects in the Pascal VOC 2007 dataset [7], a competition dataset for image retrieval. From sample images listed in Figure 14, we could find it a challenging dataset, for the significant varying of the appearances, scales, numbers and locations of the objects. A standard benchmark is provided by Pascal Challenge, calculating the Average Precision (AP) measure.

The step size and scale of patches for SIFT descriptors are 6 and 4, while 7 and 12 for Edge-SIFT descriptors. We report a 53.89% precision, winning LLC [22] (49.13%) remarkably by 4.74%.

6.2 Caltech101 Dataset

The Caltech101 dataset [8] contains 9144 images of 102 categories, including a background category. There exists significant deformation among different objects from the same category. Sample images are listed in Figure 15.

Table 3: Classification results on Caltech101.

# training	5	10	15	20	30
Lazebnik [12]	-	-	56.4	-	64.6
Yang [23]	-	-	67.0	-	73.2
Wang [22]	51.15	59.77	65.43	67.74	73.44
Boureau [3]	-	-	-	-	75.7
Bosch [2]	-	-	-	-	81.3
Ours	**61.90**	**71.75**	**76.03**	**78.53**	**82.45**
	±0.54	±0.60	±0.63	±0.39	±0.59

Figure 16: Sample images from Caltech256 dataset. Upper: categories on which accuracies are 90% or higher (30 trains). Middle: images with small objects. Bottom: images with many objects.

In this dataset, we use 7×7 SIFT patches and 12×12 Edge-SIFT patches, and the step sizes are both fixed to 7. We have used 5, 10, 15, 20, 30 images per category for training, and others for testing. Results are concluded in Table 3. In all cases, GPP outperformed LLC by more than 9%, or even more than 10% in scenarios of fewer training samples.

6.3 Caltech256 Dataset

The Caltech256 dataset [10] contains 30607 images of 257 categories, including a clutter category. It is much more challenging than Caltech101, for the objects in it not only suffer from significant size changes and deformation, but are also located on different positions on images. Figure 16 shows some sample images for this dataset.

In this dataset, 5×5 SIFT patches and 8×8 Edge-SIFT patches are extracted with a fixed step size of 5. We have used 5, 15, 30, 45, 60 images per category for training, and others for testing. Results are listed in Table 6.3.

6.4 15-scene Dataset

The 15-scene dataset [12] contains 15 scenes and 4485 images. It might be the most widely used dataset for scene understanding tasks, in which we need to discriminate various categories of scenes listed in Figure 17.

Table 4: Classification results on Caltech256.

# training	5	15	30	45	60
Griffin [10]	-	28.3	34.1	-	-
Wang [22]	-	34.36	41.19	45.31	47.68
Bosch [2]	-	-	44.0	-	-
Ours	**26.12**	**36.35**	**45.07**	**48.02**	**50.33**
	±0.21	±0.31	±0.24	±0.25	±0.18

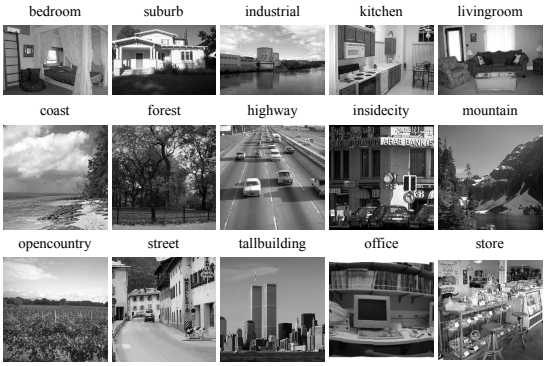

Figure 17: Sample images from 15-scene dataset

Table 5: Results on 15-scenes dataset.

# training	10	20	30	50	100
Lazebnik [12]	-	-	-	-	81.4
Wang [22]	66.97	72.44	75.78	78.84	82.34
Ours	**70.67**	**76.12**	**78.74**	**81.72**	**85.13**
	±0.46	±0.73	±0.88	±0.48	±0.72

5×5 patches with step size 5 are extracted for SIFT, and 8×8 patches with step size 8 are used for Edge-SIFT extraction. 10, 20, 30, 50, 100 images per category are used for training. Results are shown in Table 5.

7. CONCLUSIONS AND FUTURE WORK

In this paper, we propose a novel framework for image representation, and apply it for various image applications. We claim three major contributions. First, we extract SIFT and Edge-SIFT descriptors from original images and the corresponding edgemaps respectively, and then mix them together to train a joint BoF model. Experiments have revealed good compensation property of SIFT and Edge-SIFT descriptors. On the other hand, fusing them at very early stage gives us more room for spatial modeling. Second, we propose a novel pooling strategy named Geometric Phrase Pooling. By constructing geometric visual phrases upon complementary visual words, it is possible to model spatial structures containing both texture and shape features, providing more robust mid-level representation. Third, based on edge detection, we propose a simple and effective spatial weighting scheme to detect Regions of Interest. Integrating all the above coherently, we obtain a very powerful model that outperforms state-of-the-art algorithms on image classifications, retrieval and understanding applications.

However, there are still some open problems in our framework. As we have seen, objects are better described by different heterogeneous features such as texture and shape. If we could integrate heterogeneous features in a coherent way, it is possible to obtain a more discriminative description of different objects. When we are constructing spatial structures, we simply group adjacent visual words together, ignoring the size and orientation of local descriptors, which

could be useful for sophisticated spatial modeling. Also, the information contained in edgemaps is not completely exploited, and our spatial weighting scheme based on region of interest is still naive. For these problems, we might seek help from Computational Geometric algorithms for better understanding of the detected edgemaps. We will investigate these problems in our future work.

8. ACKNOWLEDGMENTS

This work was supported by the National Basic Research Program (973 Program) of China under Grant 2012CB316301, and Basic Research Foundation of Tsinghua National Laboratory for Information Science and Technology (TNList). This work was also supported in part to Dr. Qi Tian by ARO grant W911NF-12-1-0057, NSF IIS 1052851, Faculty Research Awards by Google, NEC Laboratories of America and FXPAL, UTSA START-R award, respectively.

9. REFERENCES

[1] A. Bosch, A. Zisserman, and X. Munoz. Scene Classification via pLSA. *International Conference on Computer Vision*, pages 517 – 530, 2006.

[2] A. Bosch, A. Zisserman, and X. Muoz. Image Classification using Random Forests and Ferns. *International Conference on Computer Vision*, pages 1 – 8, 2007.

[3] Y. Boureau, F. Bach, Y. LeCun, J. Ponce, et al. Learning Mid-Level Features for Recognition. *Computer Vision and Pattern Recognition*, pages 2559 – 2566, 2010.

[4] J. Canny. A Computational Approach to Edge Detection. *Pattern Analysis and Machine Intelligence*, (6):679 – 698, 1986.

[5] C. Chang and C. Lin. LibSVM: A Library for Support Vector Machines. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2(3):27, 2011.

[6] N. Dalal and B. Triggs. Histograms of Oriented Gradients for Human Detection. *Computer Vision and Pattern Recognition*, 1:886 – 893, 2005.

[7] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results. 2007.

[8] 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. *Computer Vision and Image Understanding*, 106(1):59 – 70, 2007.

[9] J. Feng, B. Ni, Q. Tian, and S. Yan. Geometric Lp-norm Feature Pooling for Image Classification. *Computer Vision and Pattern Recognition*, pages 2609 – 2704, 2011.

[10] G. Griffin, A. Holub, and P. Perona. Caltech-256 Object Category Dataset. 2007.

[11] L. Itti, C. Koch, and E. Niebur. A Model of Saliency-Based Visual Attention for Rapid Scene Analysis. *Pattern Analysis and Machine Intelligence*, 20(11):1254 – 1259, 1998.

[12] S. Lazebnik, C. Schmid, and J. Ponce. Beyond Bags of Features: Spatial Pyramid Matching for Recognizing Natural Scene Categories. *Computer Vision and Pattern Recognition*, 2:2169 – 2178, 2006.

[13] H. Lee, A. Battle, R. Raina, and A. Ng. Efficient Sparse Coding Algorithms. *Neural Information Processing Systems*, 19:801, 2007.

[14] D. Li, L. Yang, X. Hua, and H. Zhang. Large-Scale Robust Visual Codebook Construction. In *ACM Multimedia*, pages 1183–1186, 2010.

[15] D. Liu, G. Hua, P. Viola, and T. Chen. Integrated Feature Selection and Higher-Order Spatial Feature Extraction for Object Categorization. In *Computer Vision and Pattern Recognition*, pages 1–8, 2008.

[16] D. G. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. *Internatial Journal of Computer Vision*, 60(2):91 – 110, 2004.

[17] Y. Lu, L. Zhang, J. Liu, and Q. Tian. Constructing Lexica of High-level Concepts with Small Semantic Gap. *IEEE Transactions on Multimedia*, 12(4), 2010.

[18] J. Matas, O. Chum, M. Urban, and T. Pajdla. Robust Wide-Baseline Stereo from Maximally Stable Extremal Regions. *Image and Vision Computing*, 22(10):761 – 767, 2004.

[19] M. Ruzon and C. Tomasi. Color Edge Detection with the Compass Operator. *Computer Vision and Pattern Recognition*, 2, 1999.

[20] Q. Tian, S. Zhang, W. Zhou, R. Ji, B. Ni, and N. Sebe. Building Descriptive and Discriminative Visual Codebook for Large-scale Image Applications. *Multimedia Tools and Applications*, 51(2):441 – 477, 2011.

[21] A. Vedaldi and B. Fulkerson. VLFeat: An Open and Portable Library of Computer Vision Algorithms. *ACM Multimedia*, pages 1469 – 1472, 2010.

[22] J. Wang, J. Yang, K. Yu, F. Lv, T. Huang, and Y. Gong. Locality-Constrained Linear Coding for Image Classification. *Computer Vision and Pattern Recognition*, pages 3360 – 3367, 2010.

[23] J. Yang, K. Yu, Y. Gong, and T. Huang. Linear Spatial Pyramid Matching using Sparse Coding for Image Classification. *Computer Vision and Pattern Recognition*, pages 1794 – 1801, 2009.

[24] J. Yuan, Y. Wu, and M. Yang. Discovery of Collocation Patterns: from Visual Words to Visual Phrases. *Computer Vision and Pattern Recognition*, pages 1 – 8, 2007.

[25] J. Yuan, M. Yang, and Y. Wu. Mining Discriminative Co-occurrence Patterns for Visual Recognition. *Computer Vision and Pattern Recognition*, pages 2777 – 2784, 2011.

[26] S. Zhang, Q. Huang, G. Hua, S. Jiang, W. Gao, and Q. Tian. Building Contextual Visual Vocabulary for Large-Scale Image Applications. In *ACM Multimedia*, pages 501 – 510, 2010.

[27] S. Zhang, Q. Tian, G. Hua, Q. Huang, and S. Li. Descriptive Visual Words and Visual Phrases for Image Applications. In *ACM Multimedia*, pages 75–84, 2009.

[28] Y. Zhang, Z. Jia, and T. Chen. Image Retrieval with Geometry-Preserving Visual Phrases. In *Computer Vision and Pattern Recognition*, pages 809–816, 2011.

[29] W. Zhou, Y. Lu, H. Li, Y. Song, and Q. Tian. Spatial Coding for Large Scale Partial-Duplicate Web Image Search. In *ACM Multimedia*, pages 511 – 520, 2010.