A Fast Algorithm for Creating a Compact and Discriminative Visual Codebook

Lei Wang[1], Luping Zhou[1], and Chunhua Shen[2]

[1] RSISE, The Australian National University, Canberra ACT 0200, Australia
[2] National ICT Australia (NICTA)*, Canberra ACT 2601, Australia

Abstract. In patch-based object recognition, using a compact visual codebook can boost computational efficiency and reduce memory cost. Nevertheless, compared with a large-sized codebook, it also risks the loss of discriminative power. Moreover, creating a compact visual codebook can be very time-consuming, especially when the number of initial visual words is large. In this paper, to minimize its loss of discriminative power, we propose an approach to build a compact visual codebook by maximally preserving the separability of the object classes. Furthermore, a fast algorithm is designed to accomplish this task effortlessly, which can hierarchically merge 10,000 visual words down to 2 in ninety seconds. Experimental study shows that the compact visual codebook created in this way can achieve excellent classification performance even after a considerable reduction in size.

1 Introduction

Recently, patch-based object recognition has attracted particular attention and demonstrated promising recognition performance [1,2,3,4]. Typically, a visual codebook is created as follows. After extracting a large number of local patch descriptors from a set of training images, k-means or hierarchical clustering is often used to group these descriptors into n clusters, where n is a predefined number. The center of each cluster is called "visual word", and a list of them forms a "visual codebook". By labelling each descriptor of an image with the most similar visual word, this image is characterized by an n-dimensional histogram counting the number of occurrences of each word. The visual codebook can have critical impact on recognition performance. In the literature, the size of a codebook can be up to 10^3 or 10^4, resulting in a very high-dimensional histogram.

A compact visual codebook has advantages in both computational efficiency and memory usage. For example, when linear or nonlinear SVMs are used, the complexity of computing the kernel matrix, testing a new image, or storing the support vectors is all proportional to the codebook size, n. Also, many algorithms working well in a low dimensional space will encounter difficulties such

* National ICT Australia is funded by the Australian Government's *Backing Australia's Ability* initiative, in part through the Australian Research Council. The authors thank Richard I. Hartley for many insightful discussions.

D. Forsyth, P. Torr, and A. Zisserman (Eds.): ECCV 2008, Part IV, LNCS 5305, pp. 719–732, 2008.

as singularity or unreliable parameter estimate when the dimensions increase. This is often called the "curse of dimensionality". A compact visual codebook provides a lower-dimensional representation and can effectively avoid these difficulties. Moreover, in patch-based object recognition, the histogram used to represent an image is essentially a discrete approximation of the distribution of visual words in that image. A large-sized visual codebook may overfit this distribution, as pointed out in [5]. Pioneering work of creating a compact and discriminative visual codebook has been seen recently in [4], which hierarchically merges the visual words in a large-sized initial codebook. To minimize the loss of discriminative ability, the work in [4] requires the new histograms to maximize the conditional probability of the true labels of training images (or image regions in their work). This is a rigorous but complicated criterion that involves non-trivial computation after each merging operation. Moreover, at each level of the hierarchy, the optimal pair of words to be merged are sought by an exhaustive search. These lead to a heavy computational load when dealing with large-sized initial codebooks.

Creating a compact codebook is essentially a dimensionality reduction problem. To preserve the discriminative power, any classification performance related criterion may be adopted, for example, the rigorous Bayes error rate, error bounds or distances, class separability measure, or that used in [4]. We pay particular interest to the class separability measure because of its simplicity and efficiency. By using this measure, we build a compact visual codebook that maximally preserves the separability of the object classes. More importantly, we propose a fast algorithm to accomplish this task effortlessly. By this algorithm, the class separability measure can be immediately evaluated once two visual words are merged. Also, searching for the optimal pair of words to be merged is cast as a 2D geometry problem and testing a small number of pairs is sufficient to find the optimal pair. Given an initial codebook of 10,000 visual words, the proposed fast algorithm can hierarchically merge them down to 2 words in ninety seconds. As experimentally demonstrated, our algorithm can produce a compact codebook which is comparable to or even better than that obtained by [4], but our algorithm needs much less computational overhead, especially when the size of the initial codebook is large.

2 The Scatter-Matrix Based Class Separability Measure

This measure involves the *Within-class scatter matrix* ($\mathbf{H}$), the *Between-class scatter matrix* ($\mathbf{B}$), and the *Total scatter matrix* ($\mathbf{T}$). Let $(\mathbf{x}, y) \in (\mathbb{R}^n \times \mathcal{Y})$ denote a training sample, where $\mathbb{R}^n$ stands for an n-dimensional input space, and $\mathcal{Y} = \{1, 2, \cdots, c\}$ is the set of c class labels. The number of samples in the i-th class is denoted by l_i. Let $\mathbf{m}_i$ be the mean vector of the i-th class and $\mathbf{m}$ be the mean vector of all classes. The scatter matrices are defined as

$$\begin{aligned} \mathbf{H} &= \textstyle\sum_{i=1}^{c} \left[\sum_{j=1}^{l_i} (\mathbf{x}_{ij} - \mathbf{m}_i)(\mathbf{x}_{ij} - \mathbf{m}_i)^\top \right] \\ \mathbf{B} &= \textstyle\sum_{i=1}^{c} l_i (\mathbf{m}_i - \mathbf{m})(\mathbf{m}_i - \mathbf{m})^\top \\ \mathbf{T} &= \textstyle\sum_{i=1}^{c} \left[\sum_{j=1}^{l_i} (\mathbf{x}_{ij} - \mathbf{m})(\mathbf{x}_{ij} - \mathbf{m})^\top \right] = \mathbf{H} + \mathbf{B} \ . \end{aligned} \tag{1}$$

A large class separability means small within-class scattering but large between-class scattering. A combination of two of them can be used as a measure, for example, $\mathrm{tr}(\mathbf{B})/\mathrm{tr}(\mathbf{T})$ or $|\mathbf{B}|/|\mathbf{H}|$, where $\mathrm{tr}(\cdot)$ and $|\cdot|$ denote the trace and determinant of a matrix, respectively. In these measures the scattering of data is evaluated through the mean and variance, which implicitly assumes a Gaussian distribution for each class. This drawback is overcome by incorporating the kernel trick and it makes the scatter-matrix based measure quite useful, as demonstrated in Kernel based Fisher Discriminate Analysis (KFDA) [6].

3 The Formulation of Our Problem

Given an initial codebook of n visual words, we aim to obtain a codebook consisting of m ($m \ll n$) visual words in the sense that when represented with these m visual words, the c object classes can have maximal separability.

Recall that with a set of visual words, a training image can be represented by a histogram which contains the number of occurrences of each word in this image. Let $\mathbf{x}^n$ ($\mathbf{x}^n \in \mathbb{R}^n$) and $\mathbf{x}^m$ ($\mathbf{x}^m \in \mathbb{R}^m$) denote the histograms when n and m visual words are used, respectively. In the following, we first discuss an ideal way of solving our problem, and show that such a way is impractical for patch-based object recognition. This motivates us to propose the fast algorithm in this paper.

Inferring m visual words from the n initial ones is essentially a dimensionality reduction problem. It can be represented by a linear transform as

$$\mathbf{x}^m = \mathbf{W}^\top \mathbf{x}^n \tag{2}$$

where $\mathbf{W}$ ($\mathbf{W} \in \mathbb{R}^{n \times m}$) is an $n \times m$ matrix. Let $\mathbf{B}^n$ and $\mathbf{T}^n$ denote the *between*-class and *total*-class scatter matrices when the training images are represented by $\mathbf{x}^n$. The optimal linear transform, $\mathbf{W}^\star$, can be expressed as

$$\mathbf{W}^\star = \arg \max_{\mathbf{W} \in \mathbb{R}^{n \times m}} \frac{\mathrm{tr}(\mathbf{W}^\top \mathbf{B}^n \mathbf{W})}{\mathrm{tr}(\mathbf{W}^\top \mathbf{T}^n \mathbf{W})}. \tag{3}$$

Note that the determinant-based measure is not adopted because n is often much larger than the number of training images, making $|\mathbf{B}^n|$ and $|\mathbf{T}^n|$ zero. The problem in (3) has been studied in [7] recently[1]. The optimal $\mathbf{W}$ is located by solving a series of Semi-Definite Programming (SDP) problems. Nevertheless, this SDP-based approach quickly becomes intractable when n exceeds 100, which is far less than the number encountered in practical object recognition. Moreover, the $\mathbf{W}$ in patch-based object recognition may have the following constraints:

1. $\mathbf{W}_{ij} \in \{0, 1\}$ if requiring the m new visual words to have meaningful and determined content;[2]

[1] Note that this problem is not simply the Fisher Discriminant Analysis problem. Please see [7] for the details.

[2] For example, when discriminating *motorbikes* from *airplanes*, the content of a visual word will be "handle bar" and/or "windows" rather than 31% handle bar, 27% windows, and 42% something else.

2. $\sum_{j=1}^{m} \mathbf{W}_{ij} = 1$ if requiring that each of the n visual words *only be* assigned to one of the m visual words.
3. If no words are to be discarded, the constraint of $\sum_{i=1}^{n} \mathbf{W}_{ij} \geq 1$ will be imposed because each of the n visual words *must be* assigned to one of the m visual words;

This results in a large-scale integer programming problem. Efficiently and optimally solving it may be difficult for the state-of-the-art optimization techniques. In this paper, we adopt a suboptimal approach that hierarchically merges two words while maximally maintaining the class separability at each level.

4 A Fast Algorithm of Hierarchically Merging Visual Words

To make the hierarchical merging approach efficient, we need: i) Once two visual words are merged, the resulting class separability can be quickly evaluated; ii) In searching for the best pair of words to merge, the search scope has to be as small as possible. In the following, we show how these requirements are achieved with the scatter-matrix based class separability measure.

4.1 Fast Evaluation of Class Separability

Let $\mathbf{x}_i^t = [x_{i1}^t, \cdots, x_{it}^t]$ $(i = 1, \cdots, l)$ be the i-th training image when t visual words are used, where t $(t = n, n-1, \cdots, m)$ indicates the current level in the hierarchy. Let $\mathbf{K}^t$ be the Gram matrix defined by $\{\mathbf{K}^t\}_{ij} = \langle \mathbf{x}_i^t, \mathbf{x}_j^t \rangle$. Let $\mathbf{K}_{rs}^{t-1}$ be the resulting Gram matrix after merging the r-th and s-th words at level t. Their relationship is derived as

$$\begin{aligned} \{\mathbf{K}_{rs}^{t-1}\}_{ij} &= \langle \mathbf{x}_i^{t-1}, \mathbf{x}_j^{t-1} \rangle = \textstyle\sum_{k=1}^{t-1} x_{ik}^{t-1} x_{jk}^{t-1} \\ &= \textstyle\sum_{k=1}^{t} x_{ik}^t x_{jk}^t - x_{ir}^t x_{jr}^t - x_{is}^t x_{js}^t + (x_{ir}^t + x_{is}^t)(x_{jr}^t + x_{js}^t) \\ &= \textstyle\sum_{k=1}^{t} x_{ik}^t x_{jk}^t + x_{ir}^t x_{js}^t + x_{is}^t x_{jr}^t \\ &= \{\mathbf{K}^t\}_{ij} + \{\mathbf{A}_{rs}^t\}_{ij} + \{\mathbf{A}_{rs}^t\}_{ji} \end{aligned} \tag{4}$$

where $\mathbf{A}_{rs}^t$ is a matrix defined as $\mathbf{A}_{rs}^t = \mathbf{X}_r^t (\mathbf{X}_s^t)^\top$, where $\mathbf{X}_r^t$ is $[x_{1r}^t, \cdots, x_{lr}^t]$. Hence, it can be obtained that

$$\mathbf{K}_{rs}^{t-1} = \mathbf{K}^t + \mathbf{A}_{rs}^t + (\mathbf{A}_{rs}^t)^\top. \tag{5}$$

A similar relationship exists between the class separability measures at t and $t-1$ levels. Let $\mathbf{B}^{t-1}$ and $\mathbf{T}^{t-1}$ be the matrices $\mathbf{B}$ and $\mathbf{T}$ computed with $\mathbf{x}^{t-1}$. It can be proven (the proof is omitted) that for a c-class problem,

$$\mathrm{tr}(\mathbf{B}_{rs}^{t-1}) = \sum_{i=1}^{c} \frac{\mathbf{1}^\top \mathbf{K}_{rs,i}^{t-1} \mathbf{1}}{l_i} - \frac{\mathbf{1}^\top \mathbf{K}_{rs}^{t-1} \mathbf{1}}{l}; \quad \mathrm{tr}(\mathbf{T}_{rs}^{t-1}) = \mathrm{tr}(\mathbf{K}_{rs}^{t-1}) - \frac{\mathbf{1}^\top \mathbf{K}_{rs}^{t-1} \mathbf{1}}{l} \tag{6}$$

where $\mathbf{K}_{rs,i}^{t-1}$ is computed by the training images from class i. It can be verified that $\mathbf{K}_{rs,i}^{t-1} = \mathbf{K}_i^t + \mathbf{A}_{rs,i}^t + (\mathbf{A}_{rs,i}^t)^\top$. The l_i is the number of training images from

class i, and l is the total number. Note that $\mathbf{1}^\top \mathbf{A}_{rs}^t \mathbf{1} = \mathbf{1}^\top (\mathbf{A}_{rs}^t)^\top \mathbf{1}$, where $\mathbf{1}$ is a vector consisting of "1". By combining (5) and (6), we obtain that

$$\begin{aligned} \mathrm{tr}(\mathbf{B}_{rs}^{t-1}) &= \left(\sum_{i=1}^{c} \frac{\mathbf{1}^\top \mathbf{K}_i^t \mathbf{1}}{l_i} - \frac{\mathbf{1}^\top \mathbf{K}^t \mathbf{1}}{l}\right) + 2\left(\sum_{i=1}^{c} \frac{\mathbf{1}^\top \mathbf{A}_{rs,i}^t \mathbf{1}}{l_i} - \frac{\mathbf{1}^\top \mathbf{A}_{rs}^t \mathbf{1}}{l}\right) \\ &= \mathrm{tr}(\mathbf{B}^t) + 2\left(\sum_{i=1}^{c} \frac{\mathbf{1}^\top \mathbf{A}_{rs,i}^t \mathbf{1}}{l_i} - \frac{\mathbf{1}^\top \mathbf{A}_{rs}^t \mathbf{1}}{l}\right) \\ &\triangleq \mathrm{tr}(\mathbf{B}^t) + f(\mathbf{X}_r^t, \mathbf{X}_s^t), \end{aligned} \tag{7}$$

where $f(\mathbf{X}_r^t, \mathbf{X}_s^t)$ denotes the second term in the previous step. Similarly,

$$\begin{aligned} \mathrm{tr}(\mathbf{T}_{rs}^{t-1}) &= \left(\mathrm{tr}(\mathbf{K}^t) - \frac{\mathbf{1}^\top \mathbf{K}^t \mathbf{1}}{l}\right) + 2\left(\mathrm{tr}(\mathbf{A}_{rs}^t) + \frac{\mathbf{1}^\top \mathbf{A}_{rs}^t \mathbf{1}}{l}\right) \\ &= \mathrm{tr}(\mathbf{T}^t) + 2\left(\mathrm{tr}(\mathbf{A}_{rs}^t) - \frac{\mathbf{1}^\top \mathbf{A}_{rs}^t \mathbf{1}}{l}\right) \\ &\triangleq \mathrm{tr}(\mathbf{T}^t) + g(\mathbf{X}_r^t, \mathbf{X}_s^t) \quad . \end{aligned} \tag{8}$$

Since both $\mathrm{tr}(\mathbf{B}^t)$ and $\mathrm{tr}(\mathbf{T}^t)$ have been computed at level t before any merging operation, the above results indicate that to evaluate the class separability after merging two words, only $f(\mathbf{X}_r^t, \mathbf{X}_s^t)$ and $g(\mathbf{X}_r^t, \mathbf{X}_s^t)$ need to be calculated.

In the following, we further show that at any level t ($m \le t < n$), $f(\mathbf{X}_r^t, \mathbf{X}_s^t)$ and $g(\mathbf{X}_r^t, \mathbf{X}_s^t)$ can be worked out with little computation. Three cases are discussed in turn.

i) *Neither the r-th nor the s-th visual word is newly generated at level t.* This means that both of them are directly inherited from level $t+1$. Assuming that they are numbered as p and q at level $t+1$, it can be known that

$$f(\mathbf{X}_r^t, \mathbf{X}_s^t) = f(\mathbf{X}_p^{t+1}, \mathbf{X}_q^{t+1}); \tag{9}$$

ii) *Just one of the r-th and the s-th visual words is newly generated at level t.* Assume that the r-th visual word is newly generated by merging the u-th and the v-th words at level $t+1$, that is, $\mathbf{X}_r^t = \mathbf{X}_u^{t+1} + \mathbf{X}_v^{t+1}$. Furthermore, assume that $\mathbf{X}_s^t$ is numbered as q at level $t+1$. It can be shown that

$$\begin{aligned} \mathbf{A}_{rs}^t &= \mathbf{X}_r^t (\mathbf{X}_s^t)^\top = (\mathbf{X}_u^{t+1} + \mathbf{X}_v^{t+1})(\mathbf{X}_q^{t+1})^\top \\ &= \mathbf{X}_u^{t+1}(\mathbf{X}_q^{t+1})^\top + \mathbf{X}_v^{t+1}(\mathbf{X}_q^{t+1})^\top \\ &= \mathbf{A}_{uq}^{t+1} + \mathbf{A}_{vq}^{t+1}. \end{aligned} \tag{10}$$

In this way, it can be obtained that

$$\begin{aligned} & f(\mathbf{X}_r^t, \mathbf{X}_s^t) = 2\left(\sum_{i=1}^{c} \frac{\mathbf{1}^\top \mathbf{A}_{rs,i}^t \mathbf{1}}{l_i} - \frac{\mathbf{1}^\top \mathbf{A}_{rs}^t \mathbf{1}}{l}\right) \\ &= 2\left(\sum_{i=1}^{c} \frac{\mathbf{1}^\top \mathbf{A}_{uq,i}^{t+1} \mathbf{1}}{l_i} - \frac{\mathbf{1}^\top \mathbf{A}_{uq}^{t+1} \mathbf{1}}{l}\right) + 2\left(\sum_{i=1}^{c} \frac{\mathbf{1}^\top \mathbf{A}_{vq,i}^{t+1} \mathbf{1}}{l_i} - \frac{\mathbf{1}^\top \mathbf{A}_{vq}^{t+1} \mathbf{1}}{l}\right) \\ &= f(\mathbf{X}_u^{t+1}, \mathbf{X}_q^{t+1}) + f(\mathbf{X}_v^{t+1}, \mathbf{X}_q^{t+1}); \end{aligned} \tag{11}$$

iii) *Both the r-th and the s-th visual words are newly generated at level t.* This case does not exist because only one visual word can be newly generated at each level of a hierarchical clustering.

The above analysis shows that $f(\mathbf{X}_r^t, \mathbf{X}_s^t)$ can be obtained either by directly copying from level $t+1$ or by a single addition operation. All of the analysis applies to $g(\mathbf{X}_r^t, \mathbf{X}_s^t)$. Hence, once the r-th and the s-th visual words are merged, the class separability measure, $\mathrm{tr}(\mathbf{B}_{rs}^{t-1})/\mathrm{tr}(\mathbf{T}_{rs}^{t-1})$, can be immediately obtained by two addition and one division operations.

Computational complexity. The time complexity of calculating $f(\mathbf{X}_i^n, \mathbf{X}_j^n)$ or $g(\mathbf{X}_i^n, \mathbf{X}_j^n)$ is analyzed. There are $n(n-1)/2$ values to be computed in total, each of which involves computing the matrix $\mathbf{A}_{ij}^n$ which needs l^2 multiplications. Both terms of $\mathbf{1}^\top \mathbf{A}_{ij,k}^n \mathbf{1}$ $(k = 1, 2, \cdots, c)$ and $\mathbf{1}^\top \mathbf{A}_{ij}^n \mathbf{1}$ can be obtained by l^2 additions. Finally, $\sum_{i=1}^{c} (\frac{1}{l_i}) \mathbf{1}^\top \mathbf{A}_{ij,k}^n \mathbf{1} + (-\frac{1}{l}) \mathbf{1}^\top \mathbf{A}_{ij}^n \mathbf{1}$ can be worked out in $c+1$ multiplications and c additions. Hence, computing all $f(\mathbf{X}_i^n, \mathbf{X}_j^n)$ or $g(\mathbf{X}_i^n, \mathbf{X}_j^n)$ needs

$$\frac{n(n-1)}{2} \left[(l^2 + c + 1) \text{ multiplications} + (l^2 + c) \text{ additions} \right],$$

resulting in the complexity of $\mathcal{O}(n^2 l^2)$. In practice, the load of computing $\mathbf{A}_{ij}^n$ can be lower because the histogram $\mathbf{x}^n$ is often sparse. Also, $f(\mathbf{X}_i^n, \mathbf{X}_j^n)$ and $g(\mathbf{X}_i^n, \mathbf{X}_j^n)$ share the same $\mathbf{A}_{ij}^n$. The memory cost for storing all of the $f(\mathbf{X}_i^n, \mathbf{X}_j^n)$ and $g(\mathbf{X}_i^n, \mathbf{X}_j^n)$ in double precision format is $n(n-1) \times 8$ Bytes, leading to space complexity of $\mathcal{O}(n^2)$. When n equals $10,000$ (this is believed to be a reasonably large size for an initial visual codebook used in patch-based object recognition), the memory cost will be about 800 MByte, which is bearable for a desktop computer today. Moreover, the memory cost decreases quadratically with respect to the level because the total number of f or g is $t(t-1)/2$ at a given level t.

4.2 Fast Search for the Optimal Pair of Words to Merge

Although the class separability can now be quickly evaluated once a pair of words are merged, there are $\frac{t(t-1)}{2}$ possible pairs at level t from which we need to find the optimal pair to merge. If an exhaustive search is used to identify this optimal pair, the total number of pairs that are tested in the hierarchical merging process will be $\sum_{t=m+1}^{n} \frac{t(t-1)}{2}$. For $n = 10,000$ and $m = 2$, this number is as large as 1.67×10^{11}. Using an exhaustive search will significantly prolong the merging process. In the following, we propose a more efficient search strategy by making use of the properties of the scatter-matrix based class separability measure, which allows us to convert the search problem to a simple 2D geometry problem. Denote $f(\mathbf{X}_r^t, \mathbf{X}_s^t)$ and $g(\mathbf{X}_r^t, \mathbf{X}_s^t)$ by f^t and g^t in short, respectively. Recall that the class separability measure after merging two visual words is

$$\mathcal{J} = \frac{\mathrm{tr}(\mathbf{B}^{t-1})}{\mathrm{tr}(\mathbf{T}^{t-1})} = \frac{\mathrm{tr}(\mathbf{B}^t) + f^t}{\mathrm{tr}(\mathbf{T}^t) + g^t} = \frac{f^t - (-\mathrm{tr}(\mathbf{B}^t))}{g^t - (-\mathrm{tr}(\mathbf{T}^t))}$$

As illustrated in Fig. 1, geometrically, the value of $\mathcal{J}$ equals the slope of the line $\overline{AB}$ through $A(-\mathrm{tr}(\mathbf{T}^t), -\mathrm{tr}(\mathbf{B}^t))$ and $B(g^t, f^t)$.

The coordinates of A and B are restricted by the following properties of the scatter matrices:

i) From the definition in (1), it is known that

$$\mathrm{tr}(\mathbf{H}^t) \geq 0; \quad \mathrm{tr}(\mathbf{B}^t) \geq 0; \quad \mathrm{tr}(\mathbf{T}^t) = \mathrm{tr}(\mathbf{H}^t) + \mathrm{tr}(\mathbf{B}^t) \geq \mathrm{tr}(\mathbf{B}^t)$$

As a result, the point A must lie within the third quadrant of the Cartesian coordinate system gOf and above the line of $f - g = 0$. The domain of A is marked as a hatched region in Fig. 1.

ii) The coordinator of $B(g^t, f^t)$ must satisfy the following constraints:

$$\begin{aligned}
&\mathrm{tr}(\mathbf{B}^{t-1}) \geq 0 \Longrightarrow \mathrm{tr}(\mathbf{B}^t) + f^t \geq 0 \Longrightarrow f^t \geq -\mathrm{tr}(\mathbf{B}^t) \\
&\mathrm{tr}(\mathbf{T}^{t-1}) \geq 0 \Longrightarrow \mathrm{tr}(\mathbf{T}^t) + g^t \geq 0 \Longrightarrow g^t \geq -\mathrm{tr}(\mathbf{T}^t) \\
&\mathrm{tr}(\mathbf{T}^{t-1}) \geq \mathrm{tr}(\mathbf{B}^{t-1}) \Longrightarrow \mathrm{tr}(\mathbf{T}^t) + g^t \geq \mathrm{tr}(\mathbf{B}^t) + f^t \\
&\Longrightarrow f^t - g^t - (\mathrm{tr}(\mathbf{T}^t) - \mathrm{tr}(\mathbf{B}^t)) \leq 0
\end{aligned}$$

They define three half-planes in the coordinate system gOf and the point $B(g^t, f^t)$ must lie within the intersection, the blue-colored region in Fig. 1.

Therefore, finding the optimal pair of words whose combination produces the largest class separability becomes finding the optimal point $B^\star$ which maximizes the slope of the line $\overline{AB}$, where the coordinate of A is fixed at a given level t.

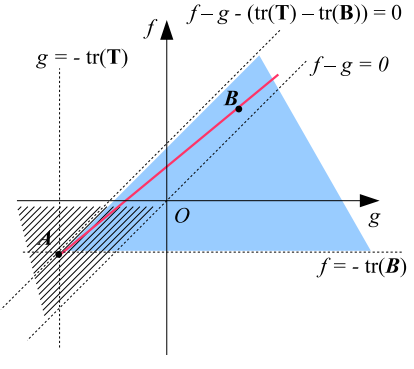


Fig. 1. Illustration of the region where $A(-\mathrm{tr}(\mathbf{T}^t), -\mathrm{tr}(\mathbf{B}^t))$ and $B(g^t, f^t)$ reside

Indexing structure. To realize the fast search, a polar coordinate based indexing structure is used to index the $t(t-1)/2$ points of $B(g, f)$ at level t, as illustrated in Fig. 2. Each point B is assigned into a bin (i,j) according to its distance from the origin and its polar angle, where $i = 1, \cdots, K$ and $j = 1, \cdots, S$. The K is the number of bins with respect to the distance from the origin, whereas S is the number of bins with respect to the polar angle. In Fig. 2, this indexing structure is illustrated by K concentric circles, each of which is further divided into S segments. The total number of bins is KS. Through this indexing structure, we can know which points B reside in a given bin. In this paper, the number of circles K is set as 40, and their radius are arranged as $r_i = r_{i+1}/2$. The S is set as 36, which evenly divides $[0, 2\pi)$ into 36 bins.

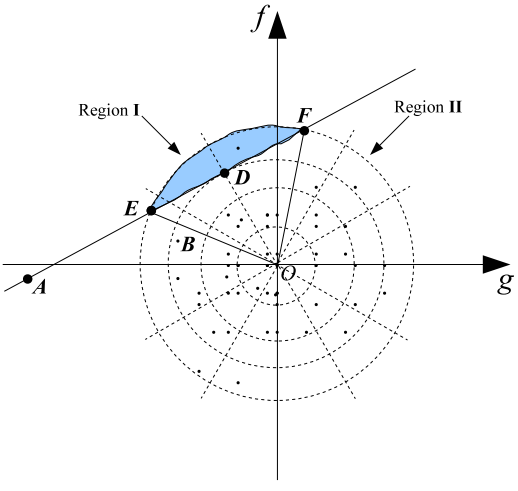


Fig. 2. The point A is fixed when searching for $B^\star$ which makes the line $\overline{AB}$ have the largest slope. The line $\overline{AD}$ is tangent to the second largest circle C_{K-1} at D, and it divides the largest circle C_K into two parts, region I and II. Clearly, a point B in region I always gives $\overline{AB}$ a larger slope than any point in region II. Therefore, if the region I is not empty, the best point $B^\star$ must reside there and searching region I is sufficient.

Search strategy. As shown in Fig. 2, let D denote the point where the line $\overline{AD}$ is tangent to the second largest circle, C_{K-1}. The line $\overline{AD}$ divides the largest circle C_K into two parts. When connected with A, a point B lying above $\overline{AD}$ (denoted by region I) always gives a larger slope than any point below it (denoted by region II). Therefore, if the region I is not empty, all points in the region II can be safely ignored. The search is merely to find the best point $B^\star$ from the region I which gives $\overline{AB}$ the largest slope. To carry out this search, we have to know which points reside in the region I. Instead of exhaustively checking each of the $\frac{t(t-1)}{2}$ points against $\overline{AD}$, this information is conveniently obtained via the above indexing structure. Let θ_E and θ_F be the polar angles of E and F where the line $\overline{AD}$ and C_K intersect. Denote the bins (with respect to the polar angle) into which they fall by S_1 and S_2, respectively. Thus, searching the region I can be accomplished by searching the bin (i, j) with $i = K$ and $j = S_1, \cdots, S_2$.[3] Clearly, the area of the searched region is much smaller than the area of C_K for moderate K and S. Therefore, the number of points $B(g, f)$ to be tested can be significantly reduced, especially when the point B distributes sparsely in the areas away from the origin. If the region I is empty, move the line $\overline{AD}$ to be tangent to the next circle, C_{K-2}, and repeat the above steps. After finding the optimal pair of words and merging them, all points $B(g, f)$ related to the two merged words will be removed. Meanwhile, new points related to the newly

[3] The region that is actually searched is slightly larger than the region I. Hence, the found best point $B^\star$ will be rejected if it is below the line $\overline{AD}$. This also means that the region I is actually empty.

generated word will be added and indexed. This process is conveniently realized in our algorithm by letting one word "absord" the other. Then, we finish the operation at level t and move to level $t-1$. Our algorithm is described in Table 1.

Before ending this section, it is worth noting that this search problem may be tackled by the dynamic convex hull [8] in computational geometry. Given the point A, the best point $B^\star$ must be a vertex of the convex hull of the points $B(g, f)$. At each level t, part of points $B(g, f)$ are updated, resulting in a dynamically changing convex hull. The technique of dynamic convex hull can be used to update the vertex set accordingly. This will be explored in future work.

Table 1. The fast algorithm for hierarchically merging visual words

Input: The l training images represented as $\{(\mathbf{x}_i, y_i)\}_{i=1}^{l}$ ($\mathbf{x}_i \in \mathbb{R}^n, y_i \in \{1, \cdots, c\}$).
The n is the size of an initial visual codebook and y_i is the class label of $\mathbf{x}_i$
m: the size of the target visual codebook.
Output: The $n - m$ level merging hierarchy

Initialization:
compute $f(\mathbf{X}_i^n, \mathbf{X}_j^n)$ and $g(\mathbf{X}_i^n, \mathbf{X}_j^n)$ $(1 \le i < j \le n)$ and store them in memory
Index the $\frac{n(n-1)}{2}$ points of $B(g, f)$ with a polar coordinate quantized into bins
Compute $A(-\mathrm{tr}(\mathbf{T}^n), -\mathrm{tr}(\mathbf{B}^n))$

Merging operation:
for $t = n, n-1, \cdots, m$
(1) **fast search** for the point $B(g^\star, f^\star)$ that gives the line $\overline{AB}$ the largest slope, where $f^\star = f^\star(\mathbf{X}_r^t, \mathbf{X}_s^t)$ and $g^\star = g^\star(\mathbf{X}_r^t, \mathbf{X}_s^t)$
(2) **compute** $\mathrm{tr}(\mathbf{B}^{t-1})$ and $\mathrm{tr}(\mathbf{T}^{t-1})$ and update the point A:
$\mathrm{tr}(\mathbf{B}^{t-1}) = \mathrm{tr}(\mathbf{B}^t) + f^\star(\mathbf{X}_r^t, \mathbf{X}_s^t)$; $\mathrm{tr}(\mathbf{T}^{t-1}) = \mathrm{tr}(\mathbf{T}^t) + g^\star(\mathbf{X}_r^t, \mathbf{X}_s^t)$
(3) **update** $f(\mathbf{X}_r^t, \mathbf{X}_i^t)$ and $g(\mathbf{X}_r^t, \mathbf{X}_i^t)$
$f(\mathbf{X}_r^t, \mathbf{X}_i^t) = f(\mathbf{X}_r^t, \mathbf{X}_i^t) + f(\mathbf{X}_s^t, \mathbf{X}_i^t)$; $g(\mathbf{X}_r^t, \mathbf{X}_i^t) = g(\mathbf{X}_r^t, \mathbf{X}_i^t) + g(\mathbf{X}_s^t, \mathbf{X}_i^t)$
remove $f(\mathbf{X}_s^t, \mathbf{X}_i^t)$ and $g(\mathbf{X}_s^t, \mathbf{X}_i^t)$
(4) **re-index** $f(\mathbf{X}_r^t, \mathbf{X}_i^t)$ and $g(\mathbf{X}_r^t, \mathbf{X}_i^t)$
end

5 Experimental Result

The proposed class separability measure based fast algorithm is tested on four classes of the Caltech-101 object database [9], including Motorbikes (798 images), Airplanes (800), Faces easy (435), and BACKGROUND_Google (520), as shown in Fig. 3. A Harris-Affine detector [10] is used to locate interest regions, which are then represented by the SIFT descriptor [11]. Other region detectors [12] and descriptors [13] can certainly be used because our algorithm has no restriction on this. The number of local descriptors extracted from the images of the four classes are about 134K, 84K, 57K, and 293K, respectively. Our algorithm is

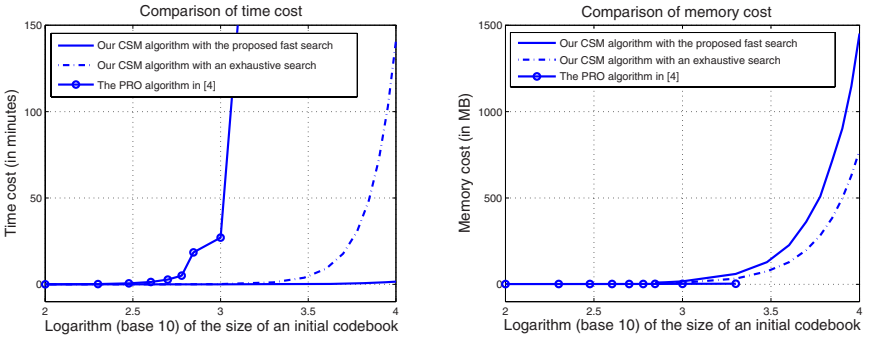
applicable to both binary and multi-class problems. This experiment focuses on the binary case, including both object categorization and object detection problems. To accumulate statistics, the images of the two object classes to be classified are randomly split as 10 pairs of training/test subsets. Restricted to the images in a training subset (those in a test subset are only used for test), their local descriptors are clustered to form the n initial visual words by using k-means clustering. Each image is then represented by a histogram containing the number of occurrences of each visual word.

Fig. 3. Example images of Motorbikes, Airplanes, Faces_easy, and BACKGROUND_Google in [9] used in this experiment

Three algorithms are compared in creating a compact visual codebook, including k-means clustering (KMS in short), the algorithm proposed in [4] (PRO in short), and our class separability measure (CSM in short) based fast algorithm. In this experiment, the k-means clustering is used to cluster the local descriptors of the training images by gradually decreasing the value of k. Its result is used as a baseline. The CSM and PRO are applied to the initial n-dimensional histograms to hierarchically merge the visual words (or equally, the bins). For each algorithm, the obtained lower-dimensional histograms are used by a classifier to separate the two object classes. Linear and nonlinear SVM classifiers with a Gaussian RBF kernel are used. Their hyper-parameters are tuned via k-fold cross-validation. The three algorithms are compared in terms of: i) the time and memory cost with respect to the number of initial visual words; ii) the recognition performance achieved by the obtained compact visual codebooks. We aim to show that our proposed CSM-based fast algorithm can achieve the recognition performance comparable to or even better than the PRO algorithm but it is much faster in creating a compact codebook.

5.1 Result on Time and Memory Cost

The time and memory cost is independently evaluated on a synthetic data set. Fixing the number of training images at 100, the size of the initial visual codebook varies between 10 and 10,000. The number of occurrences of each visual



(a) (b)

Fig. 4. Time and peak memory cost Comparison of our CSM algorithm (using the proposed fast search or an exhaustive search) and the PRO algorithm in [4]. The horizontal axis is the size (in logarithm) of an initial visual codebook, while the vertical axes are time and peak memory cost in (a) and (b), respectively. As shown, the CSM algorithm with the fast search significantly reduces the time cost for a large-sized visual codebook with acceptable memory usage.

word used in a histogram is randomly sampled from $\{0, 1, 2, \cdots, 99\}$. In this experiment, the CSM-based fast algorithm is compared with the PRO algorithm which uses an exhaustive search to find the optimal pair of words to merge. We implement the PRO algorithm according to [4], including a trick suggested to speed up the algorithm by only updating the terms related to the two words to be merged. Meanwhile, to explicitly show the efficiency of the fast search part in our algorithm, we purposely replace the fast search in the CSM-based algorithm with an exhaustive search to demonstrate the quick increase on time cost. A machine with 2.80GHz CPU and 4.0GB memory is used. The result is in Fig. 4. As seen in sub-figure(a), the time cost of the PRO algorithm goes up quickly with the increasing codebook size. It takes $1,624$ seconds to hierarchically cluster 1000 visual words to 2, whereas the CSM algorithm with an exhaustive search only uses 9 seconds to accomplish this. The less time cost is attributed to the simplicity of the CSM criterion and the fast evaluation method proposed in Section 4.1. The CSM algorithm with the fast search achieves the highest computational efficiency. It only takes 1.55 minutes to hierarchically merge 10,000 visual words to 2, and the time cost increases to 141.1 minutes when an exhaustive search is used. As shown in sub-figure(b), the price is that the fast search needs more memory (1.45GB for 10,000 visual words) to store the indexing structure. We believe that such memory usage is acceptable for a personal computer today. In the following experiments, the discriminative power of the obtained compact visual codebooks is investigated.

5.2 Motorbikes vs. Airplanes

This experiment discriminates the images of a motorbike from those containing an airplane. In each of the 10 pairs of training/test subsets, there are 959 training

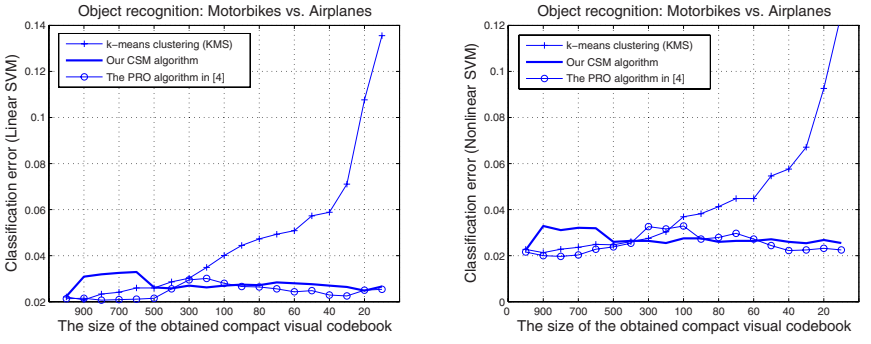


Fig. 5. Motorbikes vs. Airplanes Comparison of classification performance of the compact visual codebooks generated by k-means clustering (KMS), the PRO algorithm in [4], and our class separability measure (CSM) algorithm. Linear and nonlinear SVM classifiers are used in (a) and (b), respectively. The CSM-based algorithm still gives the excellent classification result when the codebook size has been considerably reduced.

images and 639 test images. An initial visual codebook of size $1,000$ is created by using k-means clustering. The CSM algorithm with the fast search hierarchically clusters them into 2 words in 6 seconds, whereas the PRO algorithm takes $6,164$ seconds to finish this. Based on the obtained compact visual codebook, a new histogram is created to represent each image. With the new histograms, a classifier is trained on a training subset and evaluated on the corresponding test subset. The average classification error rate is plotted in Fig. 5. The sub-figure (a) shows the result when a linear SVM classifier is used. As seen, the compact codebook generated by k-means clustering has poor discriminative power. Its classification error rate goes up with the decreasing size of the compact codebook. This is because k-means clustering uses the Euclidean distance between clusters as the merging criterion, which is not related to the classification performance. In contrast, the CSM and PRO algorithms achieve better classification performance, indicating that they well preserve the discriminative power in the obtained compact codebooks. For example, when the codebook size is reduced from 1000 to 20, these two algorithms still maintain excellent classification performance, with an increase of error rate less than 1%. Though the classification error rate of our CSM algorithm is a little bit higher (about 1.5%) at the initial stage, it soon drops to a level comparable to the error rate given by the PRO algorithm with the decreasing codebook size. Similar results can be observed from Fig. 5(b) where a nonlinear SVM classifier is employed.

5.3 Faces_Easy vs. Background_Google

This experiment aims to separate the images containing a face from the background images randomly collected from the Internet. In each training/test split, there are 100 training images and $1,498$ test images. The number of initial visual

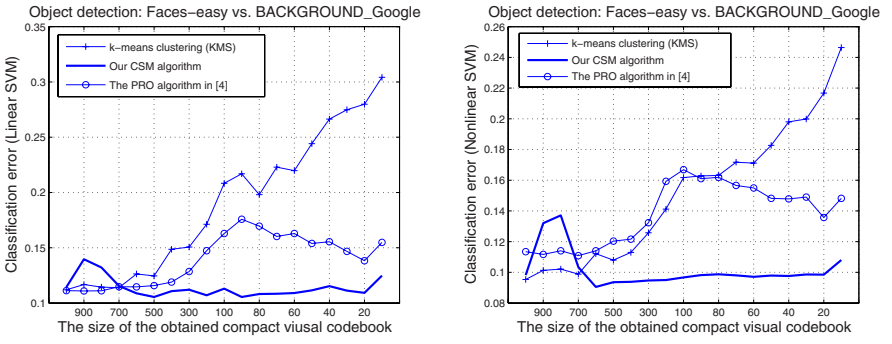


Fig. 6. Face-easy vs. Background_Google Comparison of classification performance of the small-sized visual codebooks generated by k-means clustering (KMS), the PRO algorithm in [4], and our proposed class separability measure (CSM). Linear and nonlinear SVM classifiers are used in (a) and (b), respectively. As shown, the CSM-based algorithm gives the best compact and discriminative codebooks.

words is $1,000$. They are hierarchically clustered into two words in 6 seconds by our CSM algorithm with the fast search and in $1,038$ seconds by the PRO algorithm. Again, with the newly obtained histograms, a classifier is trained and evaluated. The averaged classification error rates are presented in Fig. 6. In this experiment, the classification performance of the PRO algorithm is not as good as before. This might be caused by the hyper-parameters used in the PRO algorithm. Their values are preset according to [4] but may be task-dependent. In contrast, our CSM algorithm achieves the best classification performance. The small-sized compact codebooks consistently produce the error rate comparable to that of the initial visual codebook. This indicates that our algorithm effectively makes the compact codebooks preserve the discriminative power of the initial codebook. An additional advantage of our algorithm is that the CSM criterion is free of parameter setting. Meanwhile, a short "transition period" is observed on the CSM algorithm in Fig. 6, where the classification error rate goes up and then drops at the early stage. This interesting phenomenon will be looked into in future work.

6 Conclusion

To obtain a compact and discriminative visual codebook, this paper proposes using the separability of object classes to guide the hierarchical clustering of initial visual words. Moreover, a fast algorithm is designed to avoid a lengthy exhaustive search. As shown by the experimental study, our algorithm not only ensures the discriminative power of a compact codebook, but also makes the creation of a compact codebook very fast. This delivers an efficient tool for patch-based object recognition. In future work, more theoretical and experimental study will be conducted to analyze its performance.

References

1. Agarwal, S., Awan, A.: Learning to detect objects in images via a sparse, part-based representation. IEEE Transactions on Pattern Analysis and Machine Intelligence 26(11), 1475–1490 (2004)
2. Csurka, G., Dance, C.R., Fan, L., Willamowski, J., Bray, C.: Visual categorization with bags of keypoints. In: Proceedings of ECCV International Workshop on Statistical Learning in Computer Vision, pp. 1–22 (2004)
3. Jurie, F., Triggs, B.: Creating efficient codebooks for visual recognition. In: Proceedings of the Tenth IEEE International Conference on Computer Vision, vol. 1, pp. 604–610 (2005)
4. Winn, J., Criminisi, A., Minka, T.: Object categorization by learned universal visual dictionary. In: Proceedings of the Tenth IEEE International Conference on Computer Vision, vol. 2, pp. 1800–1807 (2005)
5. Varma, M., Zisserman, A.: A statistical approach to texture classification from single images. International Journal of Computer Vision 62(1-2), 61–81 (2005)
6. Mika, S., Rätsch, G., Weston, J., Schölkopf, B., Müller, K.R.: Fisher discriminant analysis with kernels. In: Hu, Y.H., Larsen, J., Wilson, E., Douglas, S. (eds.) Neural Networks for Signal Processing IX, pp. 41–48. IEEE, Los Alamitos (1999)
7. Shen, C., Li, H., Brooks, M.J.: A convex programming approach to the trace quotient problem. In: Yagi, Y., Kang, S.B., Kweon, I.S., Zha, H. (eds.) ACCV 2007, Part II. LNCS, vol. 4844, pp. 227–235. Springer, Heidelberg (2007)
8. Overmars, M.H., van Leeuwen, J.: Maintenance of configurations in the plane. Journal of Computer and System Sciences 23(2), 166–204 (1981)
9. Fei-Fei, L., Fergus, R., Perona, P.: Learning generative visual models from few training examples: an incremental bayesian approach tested on 101 object categories. In: Conference on Computer Vision and Pattern Recognition Workshop, vol. 12, pp. 178–178 (2004)
10. Mikolajczyk, K., Schmid, C.: Scale & affine invariant interest point detectors. International Journal of Computer Vision 60(1), 63–86 (2004)
11. Lowe, D.G.: Object recognition from local scale-invariant features. In: Proceedings of the Seventh IEEE International Conference on Computer Vision, vol. 2, pp. 1150–1157 (1999)
12. Mikolajczyk, K., Tuytelaars, T., Schmid, C., Zisserman, A., Matas, J., Schaffalitzky, F., Kadir, T., Gool, L.V.: A comparison of affine region detectors. International Journal of Computer Vision 65(1-2), 43–72 (2005)
13. Mikolajczyk, K., Schmid, C.: A performance evaluation of local descriptors. IEEE Transactions on Pattern Analysis and Machine Intelligence 27(10), 1615–1630 (2005)