IEICE TRANS. INF. & SYST., VOL.E94–D, NO.7 JULY 2011

LETTER

Quantization-Based Approximate Nearest Neighbor Search with Optimized Multiple Residual Codebooks

Yusuke UCHIDA[†a)], **Koichi TAKAGI**[†], ***and*** **Ryoichi KAWADA**[†], ***Members***

SUMMARY Nearest neighbor search (NNS) among large-scale and high-dimensional vectors plays an important role in recent large-scale multimedia search applications. This paper proposes an optimized multiple codebook construction method for an approximate NNS scheme based on product quantization, where sets of residual sub-vectors are clustered according to their distribution and the codebooks for product quantization are constructed from these clusters. Our approach enables us to adaptively select the number of codebooks to be used by trading between the search accuracy and the amount of memory available.

key words: *approximate nearest neighbor search, product quantization*

1. Introduction

Nearest neighbor search in a high-dimensional space plays an important role in many computer vision algorithms and applications, where high-dimensional feature vectors such as SIFT [1] or GIST [2] are frequently used. Given a set of data points in a metric space and a query point in the same metric space, NNS is defined as the problem of identifying the data point(s) nearest to the query point. In this paper, we focus on Euclidean space NNS, which is relevant to many applications.

The kd-tree [3] is one of the best solutions for NNS in a low-dimensional space, while its effectiveness declines as dimensionality increases due to the so-called *curse of dimensionality*. For the sake of dealing with this problem, approximate approaches such as ANN [4] or LSH [5] have attracted much attention. In approximate NNS, a search result cannot be the exact nearest neighbor point with a probability that is characterized by the parameters of approximate NNS algorithms. It is reported that a randomized kd-tree algorithm [6], [7] and a hierarchical k-means tree algorithm show better performance over ANN and LSH [8]. The randomized kd-tree algorithm constructs multiple randomized kd-trees, and these trees are explored simultaneously according to a single priority queue. The priority is determined by the distance between a query point and each bin boundary in the kd-trees (*best-bin-first*). The hierarchical k-means tree algorithm also explores the hierarchical k-means tree in a *best-bin-first* manner based on the distance between a query point and each branch node in the tree. The algorithm referred to as FLANN [8] optimally selects randomized kd-tree or hierarchical k-means tree for the indexing according to the given data distribution and the user's requirements, and it provides fully automated parameter selection. Recently, the product quantization based method has been proposed and it has been shown that it outperforms FLANN and LSH in terms of the trade-off between accuracy and search speed [9]. The most important advantage of this approach is memory efficiency. It stores only short codes created from feature vectors instead of keeping all feature vectors in the memory. It has a significant impact on large-scale multimedia search, where it is impossible to store all feature vectors in the main memory.

In addition to the product quantization based method, several methods have been proposed for generating efficient short codes: random orthogonal projection (ROP) [10], principal component analysis (PCA) [10], spectral hashing (SH) [11], and transform coding with optimized bit allocation [12]. The product quantization based method has been shown to achieve the best performance among the methods mentioned above [9], [12]. This is because these methods basically rely on scalar quantization and product quantization essentially outperforms scalar quantization in terms of the trade-off between code length and quantization error [9].

In this paper, focusing on large-scale multimedia search, an optimized multiple codebook construction method for an approximate NNS scheme based on product quantization is proposed. It enables the utilization of an arbitrary number of codebooks in product quantization, while only a single codebook can be used in the conventional scheme. Use of a larger number of codebooks increases the accuracy of NNS at the expense of increased memory requirements. In practice, one can select the number of codebooks based on the trade-off between the desired search accuracy and available memory.

2. Approximate NNS Based on Product Quantization

In this section, we briefly review the product quantization based method [9] and its problems. In this algorithm, a reference vector is decomposed into S low-dimensional subspaces and these sub-vectors are quantized separately into a short code, which is composed of their subspace quantization indices. The distance between a query vector and a reference vector is approximated by the distance between a query vector and the short code of a reference vector.

The product quantization based scheme can be integrated with an inverted index, referred to as IVFADC in [9], which enables it to avoid exhaustive searches. In IVFADC, a

Manuscript received October 8, 2010.
Manuscript revised December 27, 2010.

[†]The authors are with KDDI R&D Laboratories Inc., Fujimino-shi, 356–8502 Japan.
a) E-mail: ys-uchida@kddilabs.jp

DOI: 10.1587/transinf.E94.D.1510

reference vector is first quantized by a coarse quantizer with the size of N, then the residual vector from the corresponding centroid is encoded into a short code by product quantization. The distance between a query vector and a reference vector is approximated by the distance between the residual vector of a query vector and the short code of a reference vector. In [9], a residual sub-vector is quantized by a single codebook irrespective of the cell that each vector is quantized into, which results in increasing quantization error and degrades search accuracy, because these cells have different residual sub-vector distributions. On the other hand, if each cell has an identical codebook for product quantization for better search accuracy, the memory requirements for the codebooks become very large relative to the number of cells N in the coarse quantization. This requirement would be intractable in some situations where large N (e.g., from 10 K to 1 M in [6]) is often used. For example, assuming that $N = 100$ K, each codebook has 256 centroids and 128-dimensional SIFT features are indexed, it requires $100\,\mathrm{K} \times 256 \times 128 \sim 3$ G byte memory to store only the residual sub-vector codebooks.

3. Proposed Approach

This paper proposes the use of an arbitrary number ($1 < M < N$) of codebooks in product quantization (Fig. 1 (b)) instead of using 1 codebook or N codebooks (Fig. 1 (a) or (c)), which provides a trade-off between search accuracy and the memory requirements for the codebooks. An algorithm to create optimized codebooks for arbitrary M is also proposed, where N sets of residual sub-vectors are clustered into M clusters and M codebooks are created from these clusters. The notation used in this paper is shown in Table 1.

3.1 Multiple Codebook Construction for Product Quantization

In this section, the codebook construction scheme is described. It includes constructing the codebook C for the coarse quantization, $S \times M$ codebooks $\mathcal{D}_{s,m}$ ($1 \le s \le S$, $1 \le m \le M$) for the residual sub-vector encoding and the table $\mathrm{T_{ID}}[s][n]$ which indicates the codebook identifier $\hat{m}$, i.e., the s-th residual sub-vector in the n-th cell in the coarse quantization should be encoded by the codebook $\mathcal{D}_{s,\hat{m}}$.

Let $\mathcal{F}$ denote a set of training vectors with dimension D. We first construct the codebook C by clustering $\mathcal{F}$ into N clusters $\mathcal{F}_1, \cdots, \mathcal{F}_N$ and calculating the centroids $\mathbf{c}_1, \cdots, \mathbf{c}_N$. Then the residual vector sets $\mathcal{R}_1, \cdots, \mathcal{R}_N$ are created from $\mathcal{F}_1, \cdots, \mathcal{F}_N$ as

$$\mathcal{R}_n = \{\mathbf{f} - \mathbf{c}_n \,|\, \mathbf{f} \in \mathcal{F}_n\}. \tag{1}$$

These residual vectors are decomposed into S residual sub-vectors with dimension D/S for product quantization. Let $\mathcal{R}_{n,s}$ denote a set of s-th residual sub-vectors in the n-th cell. Our objective is to create M codebooks $\mathcal{D}_{s,1}, \cdots, \mathcal{D}_{s,M}$ for each $s \in [1, S]$ that minimize the sum of squared errors in the quantization of $\mathcal{R}_{1,s}, \cdots, \mathcal{R}_{N,s}$ using their optimal codebooks:

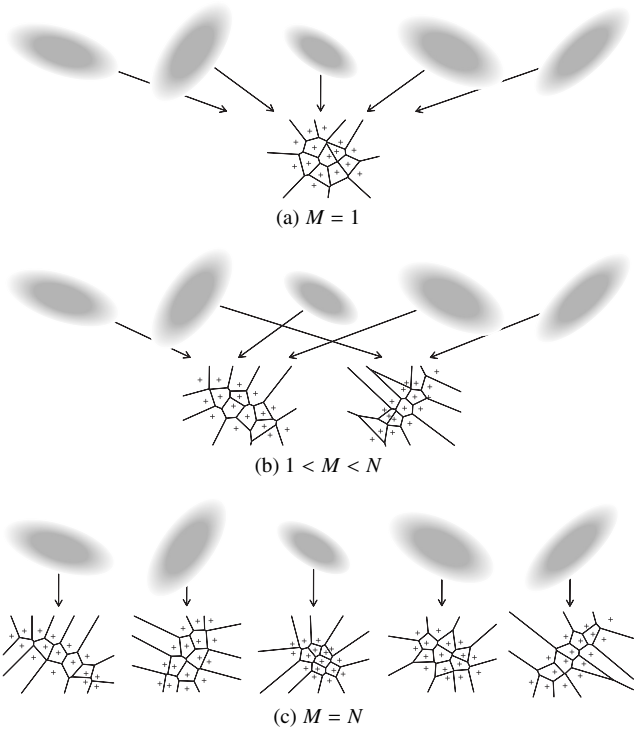
$$\text{minimize} \left(\sum_{n=1}^{N} \min_{1 \le m \le M} e(\mathcal{R}_{n,s}, \mathcal{D}_{s,m}) \right), \tag{2}$$

where $e(\mathcal{R}_{n,s}, \mathcal{D}_{s,m})$ indicates the sum of squared errors in quantizing a set of sub-vectors $\mathcal{R}_{n,s}$ using the codebook $\mathcal{D}_{s,m}$:

$$e(\mathcal{R}_{n,s}, \mathcal{D}_{s,m}) = \sum_{\mathbf{r} \in \mathcal{R}_{n,s}} \min_{\mathbf{d} \in \mathcal{D}_{s,m}} \|\mathbf{r} - \mathbf{d}\|^2. \tag{3}$$

We propose to iteratively optimize the codebooks $\{\mathcal{D}_{s,m}\}_{m=1}^{M}$ for each $s \in [1, S]$ by the following procedure. It corresponds to *clustering* $\{\mathcal{R}_{n,s}\}_{n=1}^{N}$ into M clusters via the codebooks $\{\mathcal{D}_{s,m}\}_{m=1}^{M}$.

1. Randomly select M sets of sub-vectors $\{\mathcal{R}_{f(m),s}\}_{m=1}^{M}$ out of N sets of sub-vectors $\{\mathcal{R}_{n,s}\}_{n=1}^{N}$, where $f(\cdot)$ indicates a random permutation.



(a) $M = 1$

(b) $1 < M < N$

(c) $M = N$

Fig. 1 N (= 5) residual sub-vector distributions and M codebooks.

Table 1 Notations.

D	dimension of vector space
N	codebook size of coarse quantizer
$C = \{\mathbf{c}_n\}_{n=1}^{N}$	codebook of coarse quantizer, where $\mathbf{c}_n \in \mathbb{R}^D$
S	number of residual vector decomposition
M	number of codebooks for each residual sub-vector
L	codebook size of residual sub-vector quantizer
$D_{s,m} = \{\mathbf{d}_{s,m,l}\}_{l=1}^{L}$	codebook of residual sub-vector quantizer, where $\mathbf{d}_{s,m,l} \in \mathbb{R}^{D/S}$

2. For each $m \in [1, M]$, initialize codebook $\mathcal{D}_{s,m}$ by clustering $\mathcal{R}_{f(m),s}$ into L centroids.
3. For each $n \in [1, N]$, assign a set of sub-vectors $\mathcal{R}_{n,s}$ to optimal codebook $\mathcal{D}_{s,\hat{m}}$, where $\hat{m}$ is determined by

$$\hat{m} = \operatorname*{argmin}_m e(\mathcal{R}_{n,s}, \mathcal{D}_{s,m}). \tag{4}$$

   The identifier $\hat{m}$ is also stored in the table $\mathrm{T_{ID}}[s][n]$.
4. For each $m \in [1, M]$, update codebook $\mathcal{D}_{s,m}$ by clustering all sets of sub-vectors that are assigned to $\mathcal{D}_{s,m}$ in Step 3 into L centroids.
5. Iterate Step 3 and Step 4 until $\sum_{n=1}^{N} \min_m e(\mathcal{R}_{n,s}, \mathcal{D}_{s,m})$ converges.

Codebook C, $S \times M$ codebooks $\mathcal{D}_{s,m}$ ($1 \le s \le S,\ 1 \le m \le M$) and the table $\mathrm{T_{ID}}$ created in this procedure are used in both indexing and searching procedures.

3.2 Indexing Using Multiple Codebooks

In the indexing procedure, input reference vectors are encoded into short codes and stored in the inverted index. This step is almost the same as described in [9] except for the use of multiple codebooks in residual sub-vector quantization in our scheme. First, each input vector $\mathbf{y}$ is quantized using the codebook C by

$$\hat{n} = \arg\min_n \|\mathbf{y} - \mathbf{c}_n\|^2. \tag{5}$$

Then, residual vector $\mathbf{r}$ of $\mathbf{y}$ is calculated by

$$\mathbf{r} = \mathbf{y} - \mathbf{c}_{\hat{n}}. \tag{6}$$

The residual vector $\mathbf{r}$ is divided into S sub-vectors $\mathbf{r}_1, \cdots, \mathbf{r}_S$. Each residual sub-vector $\mathbf{r}_s$ ($1 \le s \le S$) is quantized into $\hat{l}_s$ using the pre-defined codebook $\mathcal{D}_{s,\hat{m}}$, where $\hat{m} = \mathrm{T_{ID}}[s][\hat{n}]$:

$$\hat{l}_s = \arg\min_l \|\mathbf{r}_s - \mathbf{d}_{s,\hat{m},l}\|^2. \tag{7}$$

Finally, the code of S-tuple $(\hat{l}_1, \cdots, \hat{l}_S)$ is stored in the $\hat{n}$-th list of the inverted index with the vector identifier.

3.3 Approximate NNS Using Multiple Codebooks

In the search step, for each input query vector $\mathbf{x}$, the system returns the k-nearest neighbor vectors for k-NN search or the vectors with the distance less than ϵ for range search. This is performed by calculating the approximate distance $\dot{d}(\mathbf{x}, \mathbf{y})$ between query vector $\mathbf{x}$ and reference vector $\mathbf{y}$ in the inverted index. First, query vector $\mathbf{x}$ is quantized using the codebook C by

$$\hat{n} = \arg\min_n \|\mathbf{x} - \mathbf{c}_n\|^2. \tag{8}$$

The residual vector $\mathbf{r}$ of $\mathbf{y}$ is calculated and divided into S sub-vectors $\mathbf{r}_1, \cdots, \mathbf{r}_S$. Then, the distance table $\mathrm{T_{DIST}}[s][l]$ ($1 \le s \le S,\ 1 \le l \le L$) is created for subsequent distance calculations:

$$\mathrm{T_{DIST}}[s][l] = \|\mathbf{r}_s - \mathbf{d}_{s,\hat{m},l}\|^2, \tag{9}$$

where $\hat{m} = \mathrm{T_{ID}}[s][\hat{n}]$. Finally, the approximate distances between the query vector and the reference vectors in the $\hat{n}$-th list of the inverted index are calculated. The distance between the query vector $\mathbf{x}$ and the reference vector $\mathbf{y}$ with code $(l_1, \cdots, l_S)$ is efficiently calculated using the lookup table:

$$\dot{d}(\mathbf{x}, \mathbf{y}) = \sum_{s=1}^{S} \mathrm{T_{DIST}}[s][l_s]. \tag{10}$$

The search result is a set of top k-nearest reference vectors in the approximate distance for k-NN search or a set of reference vectors $\{\mathbf{y}_i\}$ that satisfies $\dot{d}(\mathbf{x}, \mathbf{y}_i) \le \epsilon$ for range search.

4. Experimental Results

4.1 Experimental Setup

In our experiments, local SIFT descriptor [1], which is one of the most frequently used features in the area of image retrieval, is used as a dataset. These SIFT descriptors are extracted from randomly selected pictures on Flickr. The dataset consists of 1,000,000 reference vectors and 10,000 query vectors. The following parameters are used in the experiments: $D = 128$, $N = 1000$, $S = 8$, $M \in [1, 1000]$ and $L = 256$.

4.2 The Impact of M on RMSE in Product Quantization

In this section, we evaluate our method in terms of root mean square error (RMSE) in optimizing the codebooks $\{\mathcal{D}_{s,m}\}$. RMSE corresponds to the upper bound of the distance between estimated distance and true distance in the searching procedure [9]. Figure 2 shows RMSE at each iteration step (calculated after Step 4) for $M = 1, 2, \cdots, 64$. Note that the norm of the SIFT descriptor is normalized to 1 here. It is found that each iteration optimizes the codebooks properly. The impact of the optimization is relatively large in case $1 \ll M \ll N$ and the optimization almost converges at the first iteration in larger M where the room for optimization

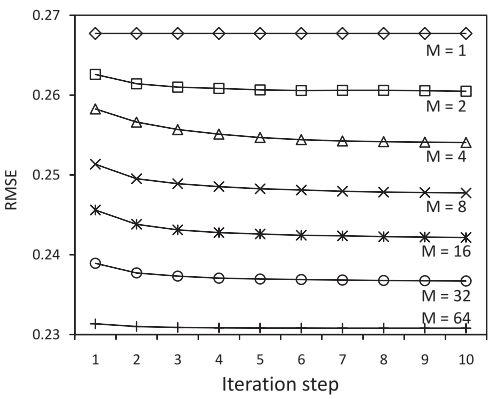


Fig. 2 Root mean squared error after each iteration.

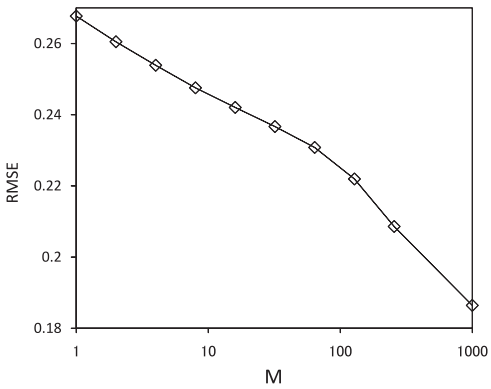


Fig. 3 Root mean squared error for each M.

is relatively small. Figure 3 shows RMSE for each M after 20 iterations. It shows the proposed scheme provides a good trade-off between RMSE and M, where M is in proportion to the amount of required memory for the codebooks $\{\mathcal{D}_{s,m}\}$. RMSE continues to decline as M becomes larger, which implies the distributions of sub-vectors in a coarse quantization cell are quite different from one another. This observation encourages a product quantization based scheme to use multiple codebooks for more accurate search results.

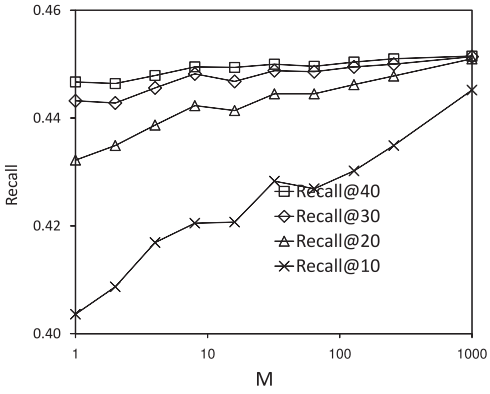
4.3 Approximate NNS Accuracy

In this section, the search accuracy of the proposed method is evaluated. The search quality is measured with recall@R in the same way as [9]. It indicates the proportion of query vectors for which the correct nearest neighbor is ranked in the first R-nearest neighbors. Figure 4 shows Recall@R for $R = 10$, 20, 30 and 40. Here w indicates the number of searched lists in the inverted index. Instead of searching the single $\hat{n}$-th list obtained from Eq. (8), w lists corresponding to the w-nearest neighbors of $\mathbf{x}$ in the coarse quantization should be searched. In this case, computational cost in the search procedure becomes w times larger, providing more accurate search results. In both cases and for each R, the proposed method provides a good trade-off between search accuracy and the memory requirements. Again, M is in proportion to the amount of required memory for the codebooks.

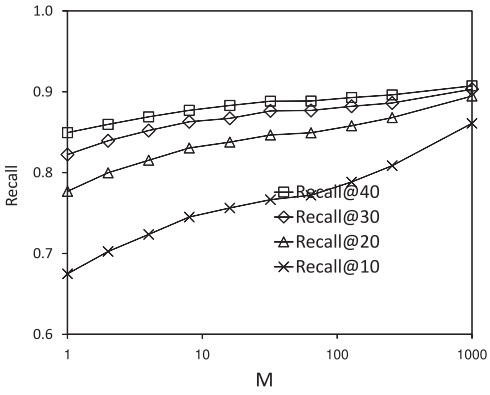
4.4 Computational Complexity

The approximate NNS with multiple codebooks has the same computational complexity as [9] in the search step. For very high-dimensional vectors, PCA is applicable as a preprocessing step to reduce computational complexity [13].

In constructing codebooks, although this is an off-line procedure, it takes much more time than the time required in constructing a single residual codebook in proportion to M (about 10 hours for $M = 256$ using a machine with a Core 2 Quad 3 GHz CPU in a single-thread program).



(a) $w = 1$



(b) $w = 10$

Fig. 4 Recall@R.

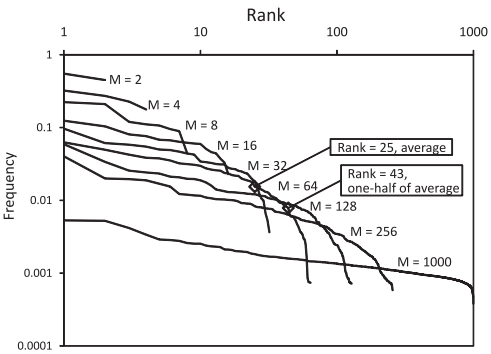


Fig. 5 Utilization rates of M codebooks.

4.5 Utilization Rates

Figure 5 shows utilization rates of M codebooks for each $M = 1, 2, \cdots, 256$, and 1000 (only the rates of codebooks for first residual subvectors ($s = 1$) are shown). Here the utilization rate of a codebook represents the proportion of residual subvectors encoded by the codebook. In the case where $M = N = 1000$, the distribution of the utilization rates correspond to the distribution of N words, which follows Zipf's law [14]. Figure 5 indicates that most codebooks are used efficiently: in the case where $M = 64$, the utilization rate of 43 out of 64 codebooks is greater than one-half

of the average (1/128), while several codebooks have very low utilization rates.

5. Conclusion

An optimized multiple codebook construction method for an approximate NNS scheme based on product quantization is proposed that enables it to use an arbitrary number of codebooks in product quantization. Experimental results show that the proposed method provides a good trade-off between search accuracy and memory requirements for the codebooks, realizing the optimization of search accuracy according to available memory resources. Future work includes the application of the proposed method to an image retrieval task and its performance evaluation.

References

[1] D.G. Lowe, "Distinctive image features from scale-invariant keypoints," Int. J. Comput. Vis., vol.60, no.2, pp.91–110, 2004.
[2] A. Oliva and A. Torralba, "Modeling the shape of the scene: A holistic representation of the spatial envelope," Int. J. Comput. Vis., vol.42, no.3, pp.145–175, 2001.
[3] J.H. Friedman, J.L. Bentley, and R.A. Finkel, "An algorithm for finding best matches in logarithmic expected time," ACM Trans. Math. Softw., vol.3, no.3, pp.209–226, 1977.
[4] S. Arya, D.M. Mount, N.S. Netanyahu, R. Silverman, and A.Y. Wu, "An optimal algorithm for approximate nearest neighbor searching in fixed dimensions," JACM, vol.45, no.6, pp.891–923, 1998.
[5] A. Andoni, "Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions," Proc. FOCS, pp.459–468, 2006.
[6] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman, "Object retrieval with large vocabularies and fast spatial matching," Proc. CVPR, 2007.
[7] C. Silpa-Anan and R. Hartley, "Optimised kd-trees for fast image descriptor matching," Proc. CVPR, 2008.
[8] M. Muja and D.G. Lowe, "Fast approximate nearest neighbors with automatic algorithm configuration," Proc. VISAPP, 2009.
[9] H. Jégou, M. Douze, and C. Schmid, "Product quantization for nearest neighbor search," IEEE Trans. Pattern Anal. Mach. Intell., vol.33, no.1, pp.117–128, 2011.
[10] H. Jégou, M. Douze, and C. Schmid, "Improving bag-of-features for large scale image search," Int. J. Comput. Vis., vol.87, no.3, pp.316–336, 2010.
[11] Y. Weiss, A. Torralba, and R. Fergus, "Spectral hashing," Proc. NIPS, 2008.
[12] J. Brandt, "Transform coding for fast approximate nearest neighbor search in high dimensions," Proc. CVPR, 2010.
[13] H. Jégou, M. Douze, C. Schmid, and P. Pérez, "Aggregating local descriptors into a compact image representation," Proc. CVPR, 2010.
[14] J. Yang, Y.G. Jiang, A.G. Hauptmann, and C.W. Ngo, "Evaluating bag-of-visual-words representations in scene classification," Proc. MIR, pp.197–206, 2007.