Distribution Sensitive Product Quantization

Linhao Li, Qinghua Hu, *Senior Member, IEEE*, Yahong Han, and Xin Li, *Fellow, IEEE*

***Abstract*— Product quantization (PQ) seems to have become the most efficient framework of performing approximate nearest neighbor (ANN) search for high-dimensional data. However, almost all existing PQ-based ANN techniques uniformly allocate precious bit budget to each subspace. This is not optimal, because data are often not evenly distributed among different subspaces. A better strategy is to achieve an improved balance between data distribution and bit budget within each subspace. Motivated by this observation, we propose to develop an optimized PQ (OPQ) technique, named distribution sensitive PQ (DSPQ) in this paper. The DSPQ dynamically analyzes and compares the data distribution based on a newly defined aggregate degree for high-dimensional data; whenever further optimization is feasible, resources such as memory and bits can be dynamically rearranged from one subspace to another. Our experimental results have shown that the strategy of bit rearrangement based on aggregate degree achieves modest improvements on most datasets. Moreover, our approach is orthogonal to the existing optimization strategy for PQ; therefore, it has been found that distribution sensitive OPQ can even outperform previous OPQ in the literature.**

***Index Terms*— Distribution sensitive product quantization (DSPQ), approximate nearest neighbor (ANN), bit allocation, aggregate degree.**

I. Introduction

NEAREST neighbor (NN) search is a fundamental tool in many computer vision and pattern recognition applications such as image classification [1]–[3], image and video retrieval/tracking [4], [5], object recognition [6] and so on. However, the computational complexity of NN search is prohibitive; in recent years, approximate nearest neighbor (ANN) search [7]–[9] has become an attractive computationally efficient alternative to NN search. The basic idea behind ANN is to *encode* the observation data into a certain number of codes. Then the actual distance between original data samples can be approximated by the distance metric calculated by their corresponding codes. Such strategy reflects the fundamental tradeoff between search performance (e.g., accuracy) and computational resources (e.g., the cost of calculation and the memory requirement).

Manuscript received February 2, 2017; revised May 23, 2017 and June 29, 2017; accepted September 18, 2017. Date of publication October 4, 2017; date of current version December 4, 2018. This work was supported by the National Natural Science Foundation under Grant 61732011. This paper was recommended by Associate Editor M. Cagnazzo. *(Corresponding author: Qinghua Hu.)*

L. Li, Q. Hu, and Y. Han are with the School of Computer Science and Technology and the Tianjin Key Laboratory of Cognitive Computing and Application, Tianjin University, Tianjin 300350, China (e-mail: huqinghua@tju.edu.cn).

X. Li is with the Lane Department of Computer Science and Electrical Engineering, West Virginia University, Morgantown, WV 26506-6109 USA.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCSVT.2017.2759277

Hash learning and vector quantization are likely to be the two most common encoding techniques to solve the ANN problem. In hash learning [10]–[14], data are encoded into short codes - known as hash codes whose length is much shorter than the dimension of original data. Hash learning is often considered sufficiently efficient for ANN [15], [16]; however, information loss is inevitable during the encoding process (i.e., when data are projected onto a low-dimensional space). In vector quantization (VQ) [17]–[19], each data sample is classified into a certain cluster based on its distance to a representative codeword (a.k.a. centroid). Then, the distance between different data samples is approximated by the distance calculated for their representative codewords. Compared with hash learning, vector quantization achieves higher accuracy in most situations; however, VQ quickly becomes impractical [19], [22]–[26] as the dimensionality of data increases (this has been known as the curse of dimensionality in the literature).

To overcome the above difficulty, Jégou et al. proposed a divide-and-conquer solution called product quantization (PQ) technique for handling high-dimensional data in 2011 [19]. The data space is first decomposed into M low-dimensional subspaces and they are concatenated by using a Cartesian product. Then data are quantized in each subspace separately; with affordable computational resources, PQ can achieve a much more optimized quantization performance than original VQ in high-dimensional space [27]–[33]. More recently, various strategies have been proposed to improve product quantizers [24], [25], [34]–[37] - e.g., Gong et al. discussed the optimization problems of the centroid in a binary space [23]; significant improvements can be observed when an optimized projection is introduced to the original data space, which led to the development of Optimized Product Quantization (OPQ) [24], [38] and its equivalent Cartesian k-means [36]; other ideas can be found in [25], [37], and [39]. As of today, OPQ developed in 2014 is considered a most effective divide-and-conquer quantizing framework on many datasets.

In all above-mentioned algorithms, bit budget is allocated *uniformly* to each subspace. However, uniform allocation of bit budget is not optimal in many cases. For example, when data in one subspace are distributed around some cluster centers and those in another subspace are scattered throughout the entire subspace, encoding them using the same code-length is no longer desirable. From a source coding perspective, it is often much easier to compress centrally-clustered data than dispersedly-distributed one; it follows a nonuniform allocation of bits would better fit these two subspaces with varying distributions. The mismatch between data distribution and bit

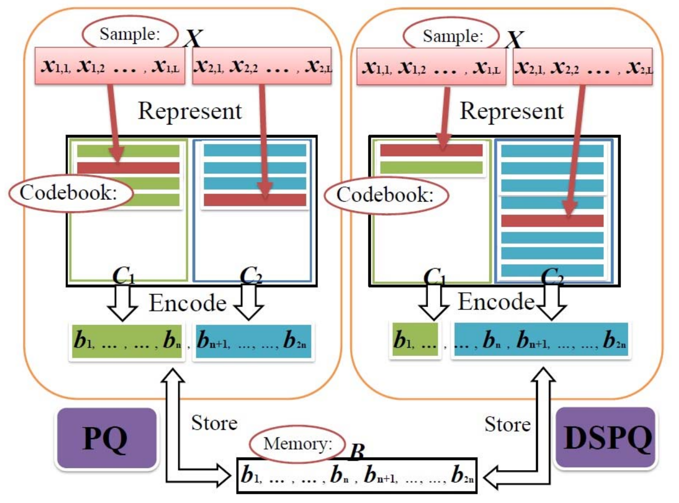


Fig. 1. Comparison between PQ (left) and DSPQ (right). Note that the method of space decomposition is the same in two competing methods whereas the strategy of bit allocation is different (therefore the sizes of the two codebooks are different).

budget of each subspace is a key problem in the previous PQ framework. It should be noted that such issue has been addressed in the pioneering work [19] where the authors claimed that each subspace should have a comparable "energy" on the average. When this assumption is violated, a strategy of multiplying the data with a random orthogonal matrix was proposed to "adjust" the natural structure of the data. Furthermore, the optimization of such orthogonal matrix was studied in OPQ [24], [38], which has resulted in state-of-the-art quantization performance.

Can we address such mismatch issue without changing the natural structure of the data? The answer is affirmative. In this paper, we propose to adjust the bit allocation of each subspace dynamically based on the varying characteristics of data distributions. In order to quantify the varying characteristics of data distributions, we have introduced a definition of aggregation degree (AD) for a considered subspace and a definition of matching index (MI) to reflect its expected quantization "energy". It can be rigorously shown that the optimal bit rearrangement can be decided based on the ratios of the corresponding matching index from the competing subspaces. Such theoretical result naturally gives rise to an iterative bit rearrangement algorithm, which is at the heart of the proposed Distribution Sensitive Product Quantization (DSPQ) method. We can also show how to solve ANN problem using DSPQ and how to integrate the strategy of bit rearrangement with existing OPQ to obtain a more powerful DS-OPQ scheme. The main contributions of this paper are summarized below:

- We propose to dynamically sense the varying characteristics of data distribution and by introducing aggregation degree. Accordingly, the quantization "energy" is represented by the matching index. A sufficient condition based on the ratio of matching indexes from different subspaces is proved for the optimality of PQ design.
- We have develop an iterative bit-rearranging algorithm to adjust the bit budget for each subspace. Such optimized allocation of bit budget based on data distribution leads to a new PQ method called DSPQ which can be easily implemented to solve ANN.
- DSPQ can also be integrated with existing OPQ. DS-OPQ can outperform OPQ (e.g., improved accuracy and faster convergence speed) without changing the natural structure of the data.

The remainder of the paper is organized as follows. We introduce some background information and motivation in Section II. The proposed technique, Distribution Sensitive Product Quantization (DSPQ) and its extension (DS-OPQ) are described in Section III. Extensive experimental results are presented in Section IV. Finally, we make several concluding remarks in Section V.

II. Background and Motivation

A. Product Quantization

A vector quantizer, denoted by Q, is a function that maps a L-dimensional vector $\mathbf{x} \in \mathcal{X}$ ($\mathcal{X}$:$\{\mathbf{x} \in \mathbb{R}^L\}$) to another vector $Q(\mathbf{x}) \in \mathcal{C}$ ($\mathcal{C}$: $\{\mathbf{c}(i), i \in \mathcal{I}\}$), where $\mathbf{c}(i)$ and $\mathcal{C}$ are known as centroid and codebook respectively. To minimize the quantization error, Q assigns an arbitrary data point $\mathbf{x}$ to its nearest centroid $\mathbf{c}(i(\mathbf{x}))$ by:

$$Q(\mathbf{x}) = \underset{\mathbf{c}(i)\in\mathcal{C}}{argmin} \|\mathbf{x} - \mathbf{c}(i(\mathbf{x}))\|, \tag{1}$$

where $i(\mathbf{x})$ is the encoder of input vector $\mathbf{x}$ and $\mathbf{c}(\cdot)$ is the decoder [17].

In the formulation of PQ [19], for an integer factor M that divides the dimension L, an L-dimensional vector can be written as the concatenation of M subvectors: $\mathbf{x} = \{\mathbf{x}_1, \cdots, \mathbf{x}_M\}$. Thus, the input space is decomposed into M subspaces accordingly and each subvector is quantized within its subspace separately. The codebook of the original space is therefore given by the concatenation of M sub-codebooks, i.e.,

$$\mathcal{C} = \mathcal{C}_1 \times \mathcal{C}_2 \times \cdots \times \mathcal{C}_M, \tag{2}$$

The training of PQ codebook is done by minimizing the following objective function:

$$\begin{aligned} &\underset{\mathcal{C}_1,\mathcal{C}_2,\ldots,\mathcal{C}_M}{min} \sum_{\mathbf{x}} \|\mathbf{x} - \mathbf{c}(i(\mathbf{x}))\| \\ &\quad s.t.\ \mathbf{c}(i(\mathbf{x})) \in \mathcal{C} = \mathcal{C}_1 \times \mathcal{C}_2 \times \cdots \times \mathcal{C}_M. \end{aligned} \tag{3}$$

where the sizes of all the M sub-codebooks are the same.

B. Optimized Product Quantization

Ge *et al.* [24], [38] proposed Optimized PQ (OPQ), which optimizes PQ by minimizing the quantization distortions with respect to the space decomposition and codebooks. By introducing a rotation matrix $\mathbf{R}$: $\{\mathbf{R}\ |\mathbf{R}^T\mathbf{R} = \mathbf{I}\}$ for the centroid $\mathbf{c}(i(\mathbf{x}))$, OPQ solves the codebook by solving the following optimization problem:

$$\begin{aligned} &\underset{\mathcal{C}_1,\mathcal{C}_2,\ldots,\mathcal{C}_M}{min} \sum_{\mathbf{x}} \|\mathbf{x} - \mathbf{c}(i(\mathbf{x}))\| \\ &\quad s.t.\ \mathbf{c}(i(\mathbf{x})) \in \mathcal{C}, \\ &\qquad \mathcal{C} = \{\mathbf{c}(i(\mathbf{x}))\ |\mathbf{R}\mathbf{c}(i(\mathbf{x})) \in \mathcal{C}_1 \times \mathcal{C}_2 \times \cdots \times \mathcal{C}_M\}. \end{aligned} \tag{4}$$

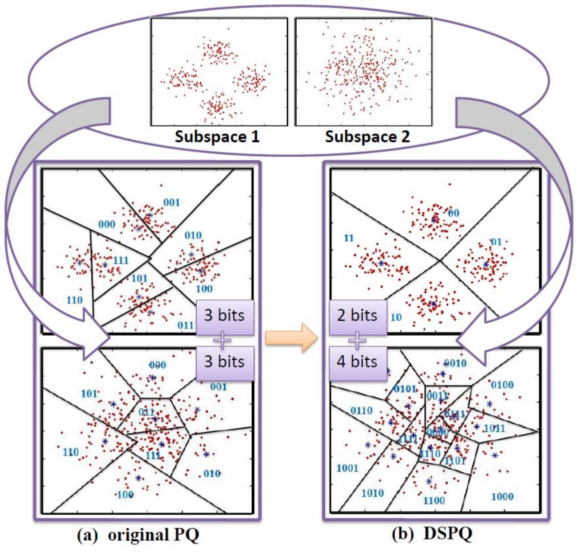


Fig. 2. Encoding manners of PQ and distribution sensitive PQ. (a) Each subspace is encoded by 3 bits in original PQ. The size of codebook is $2^3 \times 2^3 = 2^6 = 64$ (b) The first subspace is encoded by 2 bits and the second one is encoded by 4 bits. The size of corresponding codebook is $2^2 \times 2^4 = 2^6 = 64$. (Variance of Subspace1: **2.60**, Variance of Subspace2: **1.98**.)

TABLE I
QUANTIZATION DISTORTIONS OF DIFFERENT BIT-ALLOCATING WAYS

Subspace 1		Subspace 2		Whole Space	
bits	distortion	bits	distortion	centroids	distortion
1	429.05	5	23.31	64 $(2^1 \times 2^5)$	452.36
2	**90.80**	**4**	**52.23**	**64** $(\mathbf{2^2} \times \mathbf{2^4})$	**143.03**
3	59.51	3	113.22	64 $(2^3 \times 2^3)$	172.73
4	37.63	2	200.70	64 $(2^4 \times 2^2)$	238.33
5	13.57	1	393.59	64 $(2^5 \times 2^1)$	407.16

The encoding and querying procedures of OPQ are the same with those of PQ. A closely related work to OPQ is the Cartesian k-means algorithm developed by Norouzi and Fleet [36] in 2013.

C. Motivation: Analysis of Data Distribution

In the previous formulation of PQ, it is often assumed that each subspace should have a comparable "energy" on the average. When this assumption is invalid, the mismatch between data distribution and bit budget of each subspace becomes a significant issue. One way of addressing this issue is to pursue a more appropriate data representation (i.e., via matrix transforms); alternatively, one can analyze the varying characteristics of data distribution from subspace to subspace and strategically allocate the bit budget. In this paper, we advocate this alternative approach and propose to dynamically sense the data distribution for the purpose of bit rearrangement/reallocation. To facilitate our discussion of motivation, we consider a toy example as shown in Fig. 2. In this example, data points in subspace 1 form four distinct clusters and those in subspace 2 scatter across the entire space. The first quantization method follows the original PQ and uniformly allocate 3 bits to each subspace (6 bits in total); by contrast, the second method still uses 6 bits in total but one bit is donated from subspace 1 to subspace 2 (i.e., nonuniform bit allocation). Table I shows the comparison among quantization distortions of different strategies - obviously the nonuniform "2 bits + 4 bits" method achieves lower distortion than the uniform "3 bits+3 bits" method. This finding is consistent with the basic principle underlying data compression - i.e., center-clustered data (with smaller variance) are more compressible than uniform-distributed data (with larger variance).

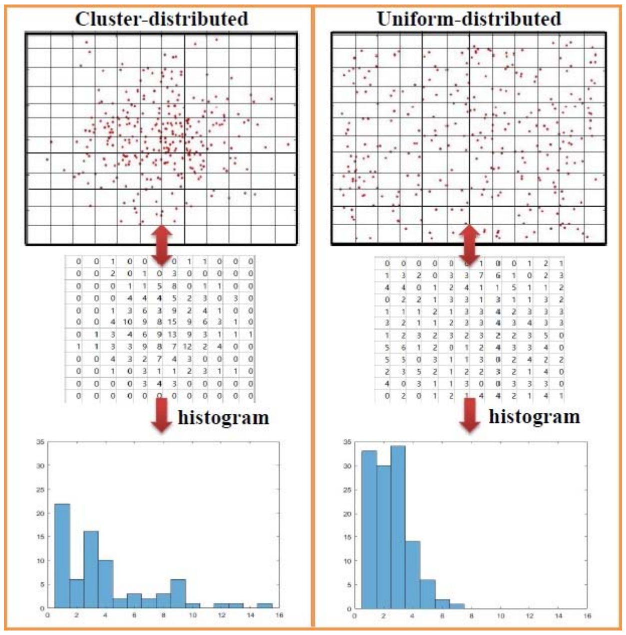


Fig. 3. Process of computing the aggregation degree of an arbitrary subspace. Top: The data space is segmented into some small cells. Here, $P = 12 \times 12 = 244$. Middle: Matrix of N_p, $p = 1, 2, \ldots, 244$. Bottom: Histogram of all the valid N_p $(N_p > 0)$.

III. DISTRIBUTION SENSITIVE PRODUCT QUANTIZATION

A. Aggregation Degree and Matching Index

We introduce some necessary notations first. For a given dataset $\mathbf{X}$ $(\mathbf{X} \in \mathbb{R}^{N \times ML})$, we assume that it is the union of M subspaces, i.e., $\mathbf{X} = [\mathbf{X}_1, \ldots, \mathbf{X}_m, \ldots, \mathbf{X}_M]$: $\mathbf{X}_m \in \mathbb{R}^{N \times L}$, where N denotes the number of samples, L denotes the dimension of data in the m-th subspace and $\mathbf{X}_m = [\mathbf{x}_1, \ldots, \mathbf{x}_l, \ldots, \mathbf{x}_L]$: $\mathbf{x}_l \in \mathbb{R}^N$. The total quantization distortion denoted by D is calculated as the sum over distortions of all subspaces - i.e., $D = D_1 + \ldots + D_M$. The total bit budget denoted by B is the sum of bit budgets for each subspace - i.e., $B = b_1 + \ldots + b_M$.

To quantify the distribution of data within a subspace, we formalize the concept of aggregation degree (AD) in low-dimension first (refer to Fig. 3). Based on the range of data samples, one can uniformly segment the data space along each dimension. For 2D toy example as shown in Fig. 3, this creates an array of quantization cells and then we can calculate the histogram (number of data points falling within each cell) for the whole data set. The result of histogram calculation will be denoted by N_p, $p = 1, 2, \ldots, P$ and only a cell with positive

N_p will be called valid (otherwise it is an empty cell). Then aggregation degree (AD) can be defined by a derivation of the standard deviation of N_p as follows

$$AD = \sum_{p=1}^{P} (N_p - \overline{N_p})^2 / (N_+)^{S_1}, S_1 \leq 1 \tag{5}$$

where $\overline{N_p}$ denotes the mean of the set $\{N_p | p = 1, 2, \ldots, P\}$ and N_+ is the total number of valid cells. S_1 is intended for shrinking the valid region, as there are usually some valid cells that contain only a few samples can hardly affect the quantization result of the entire space.

Next, we extend the above definition of AD from low-dimension to an individual subspace. Considering the m-th subspace $\mathbf{X}_m = [\mathbf{x}_1, \ldots, \mathbf{x}_l, \ldots, \mathbf{x}_L]$, we can denote the histogram calculated for the p-th cell and the l-th dimension by $N_{p,l}$. Then the aggregation degree for the m-th subspace $\mathbf{X}_m$ is given by

$$AD_m = \sum_{l=1}^{L} (\sum_{p=1}^{P} (N_{p,l} - \overline{N_{p,l}})^2 / (N_{+,l})^{S_1})^{S_2}, S_1 \leq 1, \quad S_2 \geq 1. \tag{6}$$

Here, the aggregation degree of each dimension is enlarged by S_2. Hence, the entire aggregation degree is mainly influenced by the center-cluster distributed dimensions.

It is not difficult to see that AD will be large if data are governed by a peaked Gaussian distribution (e.g., left example in Fig. 3) and will be small if the data observe a uniform distribution (right example in Fig. 3). In other words, data set with a large AD in fact would require a small bit budget (matching our intuition of more compressible) and vice versa. Therefore, to evaluate the mismatch between data distribution and bit budget for a given subspace, one can naturally define the matching index (MI) as follows:

$$V_m = AD_m \times (b_i)^{S_3}, \tag{7}$$

where b_m is the bit budget associated with subspace $\mathbf{X}_m$. S_3 penalizes the two parts to be of the same magnitude, because each dimension of AD_m is enlarged by S_2. It follows that the mismatch between two competing subspaces $\mathbf{X}_{m_1}$ and $\mathbf{X}_{m_2}$ can be characterized by their matching index difference. Intuitively, the quantization "energy" can be evaluated by the quantifying difficulty level and the sufficiency of the bits. The following lemma summarizes the key result of using matching index for the purpose of bit reallocation (its proof can be found in the Appendix).

Lemma 1: Let the matching indexes of two subspaces $\mathbf{X}_1$ *and* $\mathbf{X}_2$ *be* V_1 *and* V_2 *(*$V_1 \geq V_2$*) respectively. Then there must exist a constant* ε_* *(*$1 < \varepsilon_* < \infty$*) such that donating one bit from* $\mathbf{X}_1$ *to* $\mathbf{X}_2$ *will result in lower quantization distortion - i.e.,* $D' < D$ *whenever we have*

$$\frac{V_1}{V_2} > \varepsilon_*. \tag{8}$$

Here, D' *denotes the total quantization distortion of* $\mathbf{X}_1$ *and* $\mathbf{X}_2$ *after the bit-donating is operated,* D *denotes the distortion without bit-donating.*

An intuitive interpretation of parameter ε_* is that it serves as the lower bound of tolerable mismatch between data distribution and bit budget (conceptually similar to the role played by comparable energy in original PQ [19]). Further improvement is possible whenever some mismatch ratio bigger than ε_* can be found. Note that the parameter ε_* depends on the specific distribution of the data. In practice, we have to estimate an appropriate parameter for the datasets of our interest and carefully select this parameter empirically.

B. Formulation of DSPQ

The above theoretical analysis suggests that it is possible to improve the match between data distribution and bit budget through bit rearrangement. Bit arrangement can be done in several different ways. An ad-hoc brute force solution is to perform pairwise comparison between V_{m_1} and V_{m_2} and activate donation when Eq. (8) is satisfied; a greedy strategy would be always pick the maximum and minimum of V_m values, apply bit rearrangement until Eq. (8) is not satisfied any more. Here, we have developed an iterative matching process for bit rearrangement. First, the set $\{V_m\}$ is sorted in descending order and divided into two halves. Then we select the largest V value in the first half and denote it by $V_{*,1}$; based on $V_{*,1}$, all values in the second half satisfying inequality Eq. (8) are found. The largest of them is denoted by $V_{*,2}$ and we will donate one bit from $V_{*,1}$ to $V_{*,2}$. Such bit donation process will be repeated until the condition (8) can not be satisfied for any pair in set $\{V_i\}$. A complete description of the proposed bit rearrangement strategy is given by the following Algorithm 1.

Assuming that the optimum allocation of bits for the m-th subspace is b_m^*, obviously, we have $b_1^* + b_2^* + \cdots + b_M^* = B$. Then the codebook is

$$\mathcal{C}^{AD} = \mathcal{C}_1^{b_1^*} \times \mathcal{C}_2^{b_2^*} \cdots \times \mathcal{C}_M^{b_M^*}, \tag{9}$$

where AD represents the aggregation degree. Then, the codebook is trained by optimizing the quantization distortion.

$$\min_{\mathcal{C}^{AD}} \sum_{\mathbf{x}} \|\mathbf{x} - \mathbf{c}(i(\mathbf{x}))\|, \\ s.t. \ \mathbf{c}(i(\mathbf{x})) \in \mathcal{C}^{AD}. \tag{10}$$

Once the codebook ($\mathcal{C}_*^{AD}$) is trained, a query item $\mathbf{y}$ can be quantized by

$$Q(\mathbf{y}) = \underset{\mathbf{c}(i(\mathbf{x}))}{argmin} \|\mathbf{y} - \mathbf{c}(i(\mathbf{x}))\|, \\ s.t. \ \mathbf{c}(i(\mathbf{x})) \in \mathcal{C}_*^{AD}. \tag{11}$$

1) Time Consumption of Bit-Rearrangement: First, to calculate the matching index consumes $O(MLP^2)$. Second, the time consumption of Algorithm 1 is influenced by the data, with a maximum of $O(2^B M^2)$. These are all shorter than the time cost of training the codebook, which is $O(2^B tNL)$, where t is the target iteration number of k-means and N is the quantity of the training data.

Algorithm 1 The Bit-Rearrangement Algorithm

```
Input: $V_m, b_m, m = 1, \ldots, M$.
output: $b_m^*, m = 1, \ldots, M$.
1: re-rank the ratios $V_m$ in an increasing order get a new set:
       $\{V_m'\}, m = 1, \ldots, M$;
2: separate $\{V_m'\}$ from the median into two subsets:
       $\{V_m'^{+}\}$ and $\{V_m'^{-}\}, m = 1, \ldots, [M/2]$,
       $V_{m_1}'^{+} > V_{m_2}'^{-}$, for arbitrary $m_1$ and $m_2$;
3: while not converged do (outer loop) :
   1) $R = 0$;
   2) while not converged do (inner loop) :
      (1) $V_{*,1} = \max(V_m'^{+}), m = 1, \ldots, [M/2]$;
      (2) find all the $V_k'^{-}$ subject to:
              $V_{*,1} > \varepsilon_* \times V_k'^{-}$;
      (3) $V_{*,2} = \max(V_k'^{-})$;
      (4) if $V_{*,2} \neq \phi$
              $V_{*,1}$ and $V_{*,2}$ form the bit-donated pair,
              $R = R + 1$;
          else inner loop converged;
          end if;
      end while (inner loop);
   3) if $R > 0$
          for $m_1 = 1, \ldots, R$
            • find the subspace that corresponds to $\{V_{m_1}'^{+}\}$ and
                assume the index of the subspace is $t$;
            • $b_t = b_t - 1$;
            $m_2 = [M/2] - m_1 + 1$;
            • find the subspace that corresponds to $\{V_{m_2}'^{-}\}$ and
                assume the index of the subspace is $s$;
            • $b_s = b_s + 1$;
          end for;
      else outer loop converged;
      end if;
   end while (outer loop);
4: $b_m^* = b_m, m = 1, \ldots, M$.
```

2) Assignment Complexity of DSPQ: DSPQ only differs from PQ in terms of the way memory is allocated to each subspace. In the PQ framework, the complexity of learning the quantizer is M times the complexity of performing k-means with k^* centroids of dimension L, i.e., Mk^*L. In DSPQ, the time consumption consists of the k-means with k_m^* ($k_m^* = 2^{b_m^*}, m = 1, \ldots, M$) centroids of dimension L in all M subspaces. Then the total complexity is given by $k_1^*L + \ldots, k_M^*L$ and it is mainly influenced by the largest $k_{m\max}^*$. The relationship of the size of the codebook of both PQ (Mk^*), DSPQ ($k_1^* + \ldots, k_M^*$) and k-means (k^{*M}) is

$$k^* = 2^{b^*} < 2^{b_{m\max}^*} = k_{m\max}^* < 2^{B-M+1} < 2^B = k = k^{*M}. \tag{12}$$

Here, k is the number of centroids when clustering the data using the k-means directly. As for DSPQ, when $k_{m\max}^*$ is close to k^*, all the other k_m^* are close to k^*. Then the time consumption is also Mk^*L, which is the same as that of PQ. Otherwise, if the $k_{m\max}^*$ is larger than k^*, all the other k_m^* will be much smaller than $k_{m\max}^*$. Then the time consumption is mainly $k_{m\max}^*L$ and the time consumption of other subspaces are all several orders of magnitude smaller. $k_{m\max}^*$ is still much smaller than k. Thus, $k_{m\max}^*L$ remains much less than the time consumption of k-means, which is $Mk^{*M}L$. To summarize, the time consumption of DSPQ is usually a little more than that of PQ, but it is much less than that of k-means.

TABLE II
ASSIGNMENT COMPLEXITY PER SAMPLE AND MEMORY USAGE OF THE CODEBOOK FOR DIFFERENT QUANTIZERS

Quantizer	assignment complexity	memory usage
k-means	$MkL = Mk^{*M}L$	$MkL = Mk^{*M}L$
PQ	Mk^*L	Mk^*L
DSPQ	$(k_1^* + \ldots, k_M^*)L$	$(k_1^* + \ldots, k_M^*)L$

TABLE III
STANDARD DEVIATIONS OF AGGREGATION DEGREE VALUES THAT CORRESPOND TO THE DATA FROM ALL SUBSPACES

data	original data	initialized data
MNIST	0.3225	0.1026
SIFT1M	0.7211	0.5411
Bootstrap	1.0326	0.4588

3) Memory Usage of DSPQ: To store the codebook, DSPQ requires the memory for $k_1^* + \ldots, k_M^*$ centroids. If the values in the centroids are all floating-point values, the total cost is $(k_1^* + \ldots, k_M^*)L$ floating-point values because each centroid contains L elements. The memory usage of PQ and k-means is Mk^*L and $Mk^{*M}L$. Then, the memory usage of DSPQ is also a little more than that of PQ and is much less than that of k-means. The assignment complexity and memory usage of the codebook for different quantizers are summarized in Table II.

C. Distribution Sensitive Optimized Product Quantization (DS-OPQ)

Thanks to an improved optimization of quantization distortion, OPQ is considered to be the most effective derivative framework of PQ. However, OPQ is also a time-consuming algorithm when it is used to train a discriminative codebook. To alleviate computational burden, the authors of OPQ [24], [38] also constructed a pre-processing step based on Principal Components Analysis (PCA) for parameter initialization. Here, we want to study the impact of parameter initialization in OPQ on aggregation degree for the purpose of optimizing DSPQ. The aggregation degree values of all subspaces are collected and their standard deviations are provided in Table III. A smaller standard deviation indicates that the aggregation degree values are closer to each other after the initialization. It can be observed from Table III that the pre-processing step also helps strike an improved balance between aggregation degree and bit budget.

It is possible to further improve OPQ by employing the dynamic bit rearrangement strategy adopted by DSPQ. Without destroying the natural structure and distribution of data, we can adjust the allocation of bits dynamically

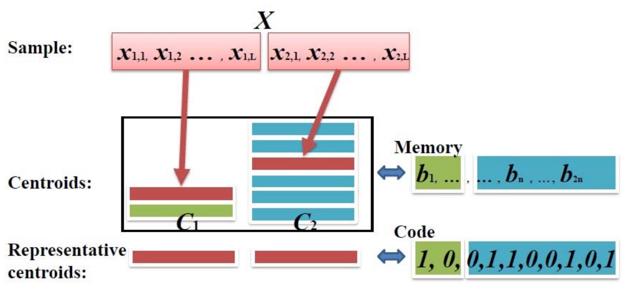


Fig. 4. The encoding process of DSPQ.

after initialization. The codebook of DS-OPQ is trained by

$$\min_{\mathbf{R},\mathcal{C}^{AD}} \sum_{\mathbf{x}} \|\mathbf{x} - \mathbf{c}(i(\mathbf{x}))\|$$
$$s.t.\ \mathbf{c}(i(\mathbf{x})) \in \mathcal{C} = \{\mathbf{c}(i(\mathbf{x}))\ |\mathbf{R}\mathbf{c}(i(\mathbf{x})) \in \mathcal{C}^{AD}\}. \quad (13)$$

The optimized codebook and rotation matrix are denoted by $\mathcal{C}_*^{AD}$ and $\mathbf{R}_*$ respectively. Here the optimizing strategies of OPQ and DS-OPQ are the same. Thus, the performance is only decided by the quantization precision or equivalently the codebook size. As shown in the experimental section, starting from an improved initialization, DS-OPQ converges much faster than OPQ.

The querying process entails solving the following problem

$$\underset{\mathbf{c}(i(\mathbf{x}))}{argmin} \|\mathbf{R}_*\mathbf{x} - \mathbf{c}(i(\mathbf{x}))\|$$
$$s.t.\ \mathbf{c}(i(\mathbf{x})) \in \mathcal{C} = \{\mathbf{c}(i(\mathbf{x}))\ |\mathbf{R}_*\mathbf{c}(i(\mathbf{x})) \in \mathcal{C}_*^{AD}\}. \quad (14)$$

D. ANN With DSPQ and DS-OPQ

ANN reduces to the task of squared distance computation once the samples are all encoded. As is shown in Fig. 4, encoding in both DSPQ and DS-OPQ means to represent a sample by M centroids whose lengths are all L. Then the entire code length is ML, which is the same as that in PQ and OPQ. Hence, the SDC and ADC proposed in [19] can be utilized to calculate the distance in DSPQ and DS-OPQ.

Besides, we can find that, although the encoding strategies of DSPQ and PQ are different, their resulting codes are of the same shapes and can be utilized in the same way. Thus, DSPQ can also be combined with an inverted files system [4]. In the system proposed in [19], we can achieve this by simply replacing the codes of PQ by the codes of DSPQ. Furthermore, the codes can be further encoded into effective binary code by the algorithm proposed in [42].

IV. Experimental Evaluation

In this section, we describe the parameter estimation, compare our algorithms to previous techniques, and further analyze our algorithms. The cell number P is set to 50, the parameters S_1 and S_2 in calculating the aggregation degree value are set to 0.5 and 1.3, respectively. S_3 is set to 1.25.

First, we introduce the datasets we included and the measurements. The public datasets include SIFT1M [19], MNIST [24], LFW [43], and Bootstrap [44]. Besides, a surveillance video from a public resource is also included and used to form a quantization task. SIFT1m and MNIST are two widely used datasets to test the PQ-based algorithms for ANN search [24], [25], [37]–[39]. Here, SIFT1M is composed of SIFT descriptors and MNIST consists of 70k handwritten digital images. LFW consists of about 1.2k colorful faces and Bootstrap contains a short video that is composed of approximately 3k frames. The actual surveillance video, which was recorded by "Surveil" is composed of 2.2k frames. Details of these datasets are given in Table IV. The samples in the training set are selected randomly from the base set.

TABLE IV
Details of the Included Datasets

dataset	dimension	train set	base set	query set
SIFT1M	128	100k	1M	10k
MNIST	28× 28	20k	60k	10k
LFW	64× 64	316	1k	200
Bootstrap	120× 160	1.5k	2.5k	500
Surveil	44× 44	700	2.2k	300

TABLE V
Bit Budget of DSPQ on SIFT1M and MNIST Datasets

task—SIFT1M	bits allocation
$M = 2$	[8, 8]
$M = 4$	[6, 10, 10, 6]
$M = 8$	[6, 7, 9, 10, 9, 10, 6, 7]
$M = 16$	[5, 8, 8, 5, 8, 11, 11, 10, 8, 11, 10, 8, 5, 7, 8, 5]
task—MNIST	bits allocation
$M = 2$	[7, 9]
$M = 4$	[4, 12, 11, 5]
$M = 8$	[2, 6, 12, 12, 11, 11, 7, 3]
$M = 16$	[2, 3, 5, 10, 11, 11, 12, 11, 10, 10, 10, 11, 10, 5, 4, 3]

Besides, three widely used criteria are included: mean Average Precision (mAP), Recall@R and Recall. mAP examines whether the algorithms can find the approximate nearest neighbors and rank them in the correct order; Recall@R is defined by the proportion of query vectors for which the nearest neighbor is ranked in the first R positions; Recall is the proportion of query vectors for which the K true Euclidean nearest neighbors are found in the first N positions. Generally, as for Recall and mAP, only the ranks of the $K = 100$ Euclidean nearest neighbors are considered.

To be more intuitively and clearly, in this paper, Recall@R and Recall are denoted by Recall(1) and Recall(100), respectively. Because Recall@R measures an algorithm with the performance of exploring the nearest neighbor, whereas Recall emphasizes the ability of finding the approximate nearest neighbors ($K = 100$). In other words, Recall@R is a specific case of Recall ($K = 1$).

Finally, as is suggested in [38], OPQ is only initialized by the parametric solution and its maximum number of iterative cycles is set as 100. In Table V, bit budget of DSPQ on SIFT1M and MNIST datasets are given. Initially, 8 bits (or 256 centroids) are assigned to each subspace.

A. Parameter Estimation

ε_* is a key threshold to determine whether the ratio of the matching indexes ε ($\varepsilon = V_1/V_2$) is obvious. Here we estimate

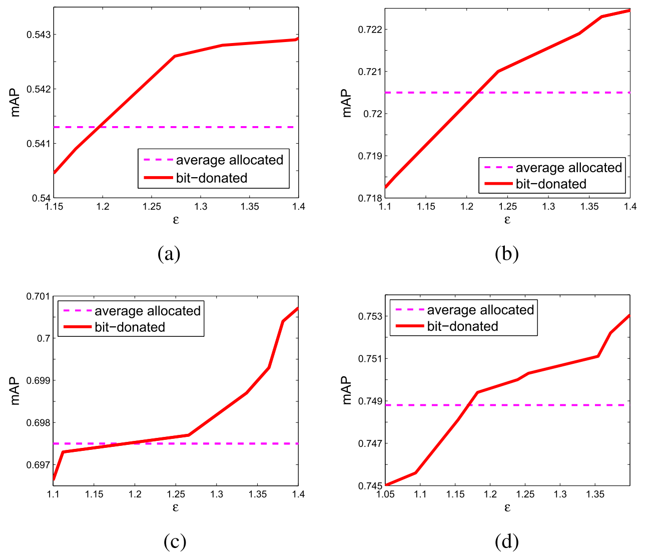


Fig. 5. Comparisons of the performances when ε changes. (a) SIFT1M. (b) MNIST. (c) Surveil. (d) Bootstrap.

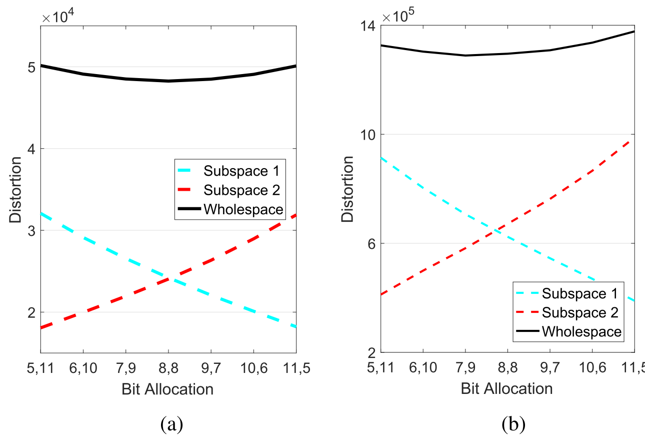


Fig. 6. Tradeoff between the number of bits and the quantization performance per subspace for different datasets when $M = 2$. (a) SIFT1M. (b) MNIST.

TABLE VI

TIME CONSUMPTION OF THE TRAINING PROCESS IN BOTH DSPQ AND PQ (THE UNIT IS SECOND)

	M	DSPQ	PQ	DSPQ-pre
SIFT1M	4	147.10	146.89	0.30
	8	209.80	199.32	0.33
	16	378.24	333.14	0.32
MNIST	4	65.33	38.86	0.26
	8	73.35	38.00	0.27
	16	74.07	38.23	0.24
Bootstrap	4	0.18	0.13	0.018
	8	0.30	0.18	0.019
	16	0.53	0.37	0.019
Surveil	4	0.16	0.15	0.036
	8	0.20	0.19	0.037
	16	0.30	0.26	0.037

the performance of the quantizers while steadily altering the value of ε. For each given value of ε, in the bit-rearranging operation, only two subspaces are included and all the others remain in the initial state. Then, one bit is donated from one subspace to another. The corresponding ratio ε and the quantization results are collected and are shown in Fig. 5. We also include the results obtained without any bit-donated operation for facilitate comparison. Actually, the crossing point of the two curves is the default value at which the two methods return the same quantization precision.

The figures show that the quantization results of the dynamically adjusted bit-allocating method is positively correlated to the parameter ε. A large ε indicates obvious difference in the aggregation degrees of the two subspaces. Thus, the dynamic bit-arrangement method definitely improves the quantization results. Otherwise, dynamic bit arrangement would be of no use if the aggregation degrees are at the same level.

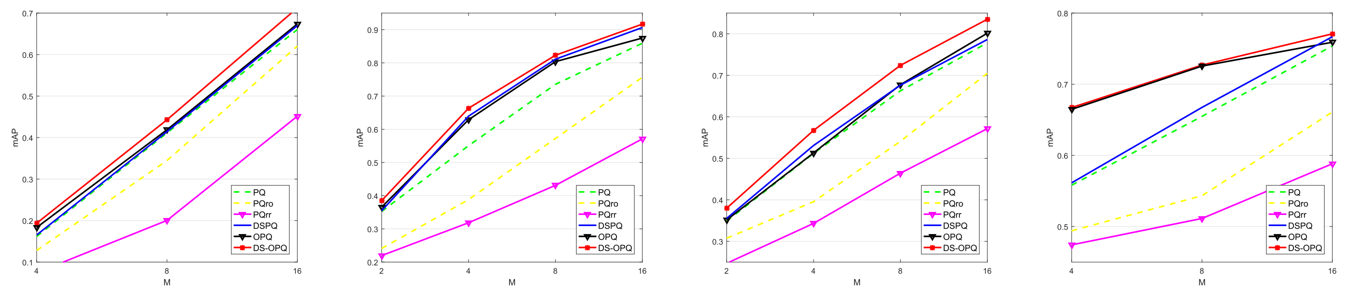
The optimal threshold ε_* relies on the specific data distribution. In Fig. 5, the intersection point of the two curves in each sub-figure varies in the range 1.17 to 1.22. Eventually, we consider $\varepsilon = 1.24$ as a cautious choice if no prior information is available. A slack threshold means that bit rearrangement only occurs when the differences in the aggregation degrees are sufficiently obvious. Otherwise, the results of DSPQ are the same as those of PQ.

B. Evaluations of DSPQ

First, we show how the bit allocation influences the quantization performance. First, the data space is decomposed into two subspaces. Then we steadily change the bit allocation for each subspace and record the corresponding quantization distortion. Eventually, the distortion of the entire space is obtained accordingly. The curves are shown in Fig. 6.

In the figure, once the memory is donated from one subspace to another, the quantization distortion increases. Meanwhile, the distortion of the other subspace decreases. Then the distortion of the entire space keeps on changing with the adjustment of the bit allocation. For different datasets, the optimized bit allocation way is determined by the data distribution. We can see from the figure that, the 7, 9-way is the best choice for the MNIST dataset, whereas the average bit allocation method is optimal for the SIFT1M dataset. When M increases, the data are segmented into additional smaller pieces. Then the distributions of the different subspaces are more flexible and the results obtained with the optimal bit allocation method differs greatly from those obtained using the average method.

Then we provide the time consumption of the preprocessing and processing operations required to train a product quantizer. The two sets of results are provided in Table VI together with the time consumed by training the product quantizer. Besides, the time consumption of the query phases are given in Table VII. We can see from the tables that both the training and the querying time costs of DSPQ are usually higher than those of PQ because DSPQ produces more centroids. In most cases, $k_1^* + \ldots, k_M^*$ is a little larger than Mk^*. Thus the time consumption of the query phase in DSPQ is almost the same as that in PQ. In the training process,



(a) (b) (c) (d)

Fig. 7. Quantization results of all the algorithms in terms of the mAP measurement. (a) SIFT1M. (b) MNIST. (c) Bootstrap. (d) Surveil.

TABLE VII
ASSIGNMENT TIME CONSUMPTION OF THE QUERY SAMPLES IN BOTH DSPQ AND PQ (THE UNIT IS SECOND)

	M	Sample	DSPQ	PQ
SIFT1M	4	10k	7601.29	7423.41
	8		14499.34	13343.16
	16		30311.63	28091.90
MNIST	4	10k	487.26	487.21
	8		889.99	875.67
	16		1655.26	1583.46
Bootstrap	4	500	1.89	1.85
	8		3.06	2.46
	16		5.33	4.85
Surveil	4	300	1.32	0.99
	8		1.76	1.63
	16		2.89	2.77

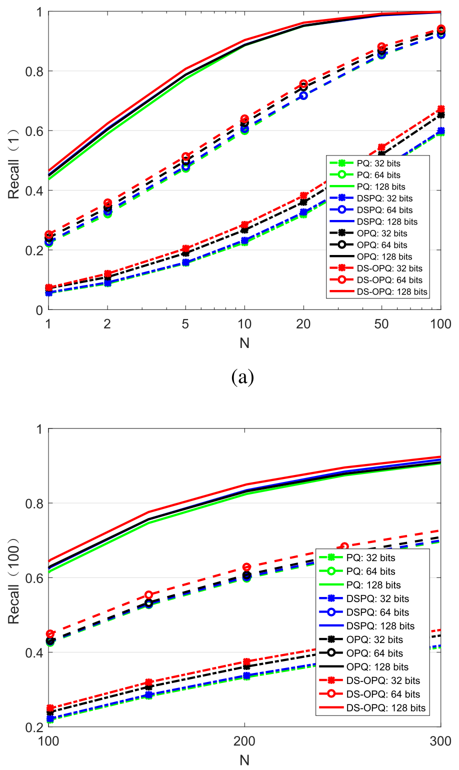
the iteration of the k-means in Matlab 2016 stops when the distortion is low enough instead of stopping after t iterations. Thus, the eventual training time of PQ is a little more than that of PQ. Besides, the pre-processing cost of DSPQ is so low that it can be ignored in the training process.

C. Comparisons of the Algorithms

We compare the performance of our algorithms with that of previous algorithms. Here, the PQ algorithm refers to the original version proposed in [19], instead of the derivative algorithms (PQ-RR and PQ-RO) in [38], because we find the original PQ to perform much more effectively than the two derivatives. The results obtained for PQ-RR and PQ-RO are also included.

The results, in terms of the mAP measurement, which estimates both the number of returned neighbors and their ranks, are shown together in Fig. 7. The task $M = 2$ is missing on some datasets. This is the case when the differences among the subspaces are smaller than the threshold, in which case no bit rearrangement is required. The results, in terms of the mAP measurement which estimates both the quantity of the returned neighbours and their ranks, are shown together in Fig. 7.

Our proposed algorithms perform more effectively than the previous ones on all the datasets, i.e., DSPQ is superior to

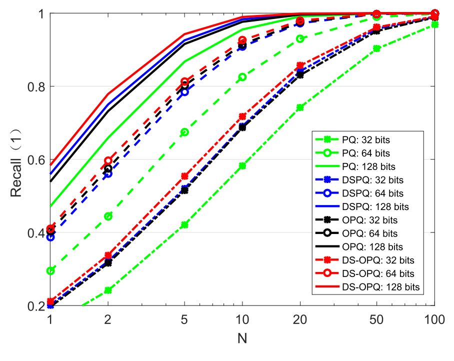


(a)

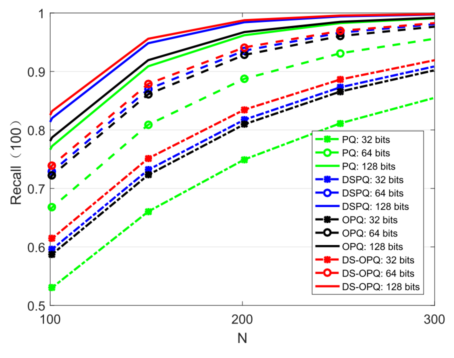
(b)

Fig. 8. Quantization results of all the algorithms on SIFT1M dataset with $M = 4$ (32 bits), 8 (64 bits), 16 (128 bits). (a) Recall(1). (b) Recall(100).

PQ and DS-OPQ outperforms OPQ, in most cases. PQ-RR and PQ-RO performed the least effectively. Thus, the approach of achieving comparable energy by multiplying the data with a random orthogonal matrix is not a good choice. Although this can balance the differences between the aggregation degree of the data and the bit budget of each subspace, it destroys the structure of the data. After the projection, selected elements that are similar or even almost the same are projected into different subspaces and are recorded by different codes. Thus, the performance of the quantizers is poor. On the contrary, the optimized approach (OPQ) results in a more effective performance because it optimizes the projection matrix and finds the best way to achieve comparable energy for each subspace. DSPQ presents another way in which to balance the differences between the aggregation degree of the data and

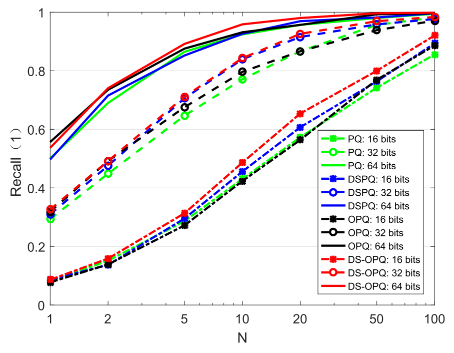


(a)

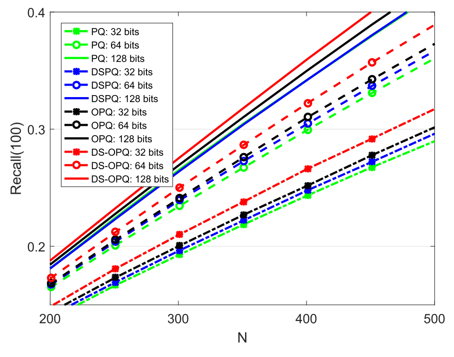


(b)

Fig. 9. Quantization results of all the algorithms on MNIST dataset with $M = 4$ (32 bits), 8 (64 bits), 16 (128 bits). (a) Recall(1). (b) Recall(100).

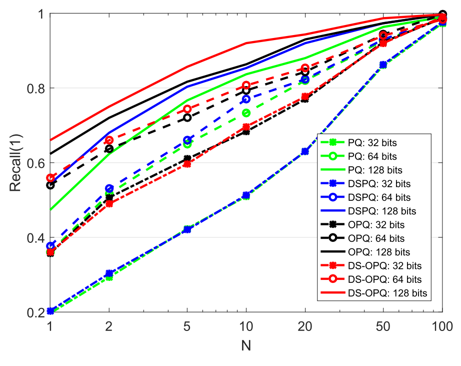


(a)

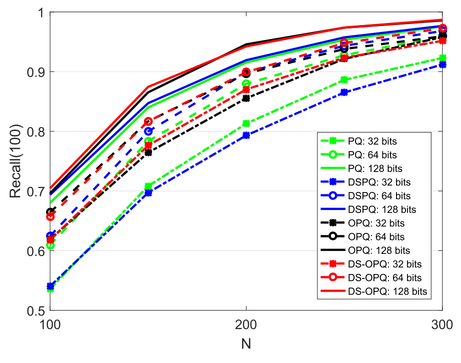


(b)

Fig. 10. Quantization results of all the algorithms on Bootstrap dataset with $M = 4$ (32 bits), 8 (64 bits), 16 (128 bits). (a) Recall(1). (b) Recall(100).



(a)



(b)

Fig. 11. Quantization results of all the algorithms on Sureil dataset with $M = 4$ (32 bits), 8 (64 bits), 16 (128 bits). (a) Recall(1). (b) Recall(100).

the bit budget of each subspace. The improvement of DSPQ compared to PQ is obvious. However, without the optimizing process, DSPQ performs less optimal than OPQ, although its time consumption is much less than OPQ. Ultimately, DS-OPQ always achieves the best performance. Compared with DSPQ, DS-OPQ is the optimized version and converges to the optimum. Compared with OPQ, DS-OPQ is more accurate because more centroids are utilized. As shown in the next section, DS-OPQ also converges faster than OPQ, because it starts from the natural structure of the data.

The advantages of the respective strategies vary on different datasets. On both SIFT1M and Bootstrap, only DS-OPQ performs much more accurately than PQ whereas OPQ and DSPQ perform at the same level as PQ. This is the case in which the use of additional centroids and further optimization are effectively combined. On the MNIST dataset, both the bit rearrangement and optimization are sufficiently effective, whereas the combination of these two strategies has limited benefit because the optima of the two cases are close to each other. On the Surveil dataset, the optimizing strategy plays a key role in improving the quantization results, whereas the bit rearrangement method results in limited improvements to both PQ and OPQ.

In summary, both the bit rearrangement and optimization strategy improve the quantization task on these datasets. The combination of the two strategies results in the most effective quantizer.

A further comparison of PQ, OPQ, DSPQ, and DS-OPQ was carried out and we provide the performance in terms of the measurements Recall(1) and Recall(100). The results of each dataset with $M = 4, 8, 16$ are presented together in the same figure. All the results are shown in Fig. 8 to Fig. 11.

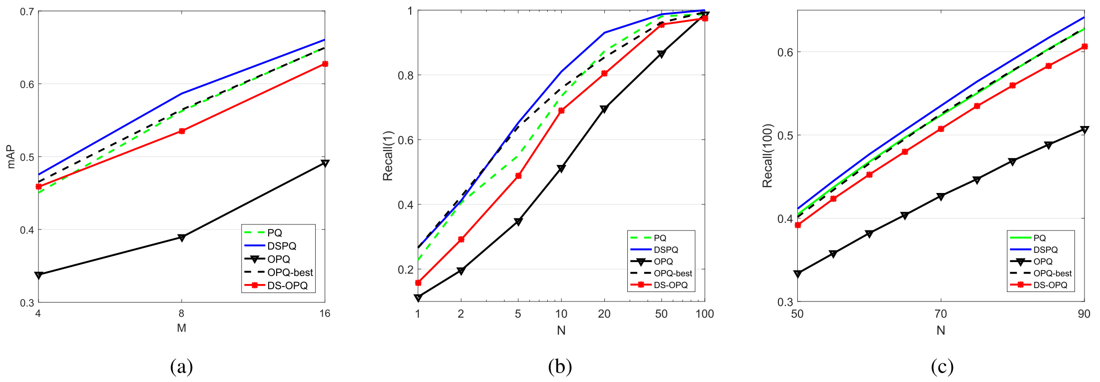


Fig. 12. Quantization results of all the algorithms on LFW dataset. (a) mAP vs. M. (b) Recall(1) (M = 8). (c) Recall(100) (M = 8).

In terms of Recall(1) and Recall(100), the results are consistent with those of mAP. Besides, the improvements vary as the number of subspaces (M) changes. The use of less memory enhances the improvements achieved with both the bit rearrangement and the optimization strategies. Otherwise, when the available memory allocated to all the subspaces is sufficient, no improvement strategy is necessary because the results of PQ are already optimal and the quantization error is close to zero. Alternatively, when the available memory is insufficient, solving the mismatch between the distribution of the data and the bit budget of each subspace is significant.

Furthermore, we also include the LFW dataset to show a case where the optimized approach is no longer effective in solving the quantization task. This is a task for which the samples are really high dimensional and similar to each other; however, the number of samples is smaller than their dimensions. This situation occurs in some practical recognition problems.

In the process of optimization, each dimension of the data is allocated a parameter. Then the training set should be sufficiently large such that the number of training samples is larger than the number of non-zero eigenvalues. Otherwise, a uniquely optimal solution would not exist and almost none of the solutions would lead to an improvement in PQ. As is shown in Fig. 12, the performance of OPQ is the least accurate in quantizing LFW data. Besides, we tried some other initializations for OPQ and found that starting from the natural structure of the data results in the best quantization performance, as shown by the black dotted line in the figure.

DSPQ achieves the best performance because it obeys the natural structure of the data, in which case the distributions of the adjacent dimensions are similar. Arranging them into the same subspace improves the quantization task. Thus, the bit rearrangement strategy continues to be effective even if the optimizing strategy is unsuccessful. Meanwhile, we find that DS-OPQ also performs poorly because the projection of the data does not lead to an improvement.

Besides, considering all the results, it is clear that DSPQ never adversely affects PQ. Because, once the bit rearrangement operation is no longer required, DSPQ uses averaging to allocate all the bits to subspaces such as PQ. Thus, the worst expectation of the performance of DSPQ is the same as that of PQ.

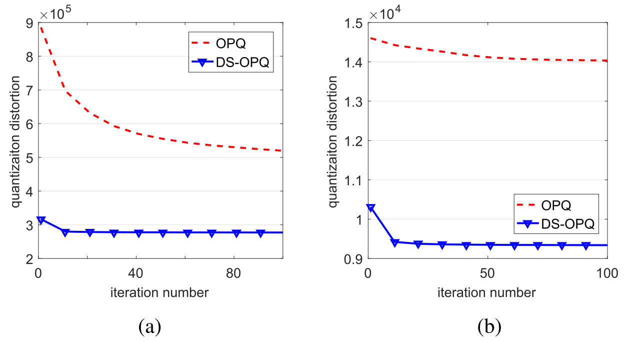


Fig. 13. The converge speed of OPQ and DS-OPQ. (a) MNIST. (b) SIFT1M.

Based on the above results, we conclude that both the proposed sensitive data distribution bit rearrangement method (DSPQ) and the optimized method (OPQ) are effective in solving most practical quantization tasks. Their combination (DS-OPQ) results in the most effective quantizer if only the optimizing strategy can find the global optimum.

D. Efficiency Estimation

We conducted a further comparison of the efficiencies of the two optimizing quantization algorithms by also demonstrating their converge speeds in this section. Although OPQ and DS-OPQ obey the same optimizing strategy, they differ from each other in terms of the starting point.

Here, as is generally accepted [19], [38], the quantization distortion is used to represent the accuracy of the quantization task. Although the quantization distortions of the two algorithms (OPQ and DS-OPQ) converge to different optima because the number of centroids differ, we are still able to judge the convergence speed of the two algorithms from the tendencies of the curves.

As is shown in Fig. 13, the objective functions of OPQ continue decreasing steadily for the duration of the 100 iterative cycles. On the MNIST dataset, the objective function of OPQ decreases sharply during the first 20 iterative cycles and its curve becomes increasingly flat in subsequent iterations. On the SIFT1M dataset, the curve of OPQ decreases slowly all the time. In summary, the curves indicate that, on both of the two datasets, OPQ converges all the time during the given iterations and the optimizing process continues even after 100 iterative cycles.

In comparison, the curves of DS-OPQ stabilize after only a few dozen cycles and the corresponding objective function remains consistent in the subsequent iterative cycles. This indicates that DS-OPQ has already converged to the optimum. Equivalently, the dynamic bit arrangement operation facilitates the original product quantization task considerably and DS-OPQ converges much faster than OPQ. Furthermore, as is shown in Section IV-C, the performance of DS-OPQ is also more effective than those of OPQ on these two datasets.

V. Conclusion and Future work

The research presented in this paper concerns the mismatch between the aggregation degree of the data and the bit budget of each subspace. Hence, we proposed a dynamic bit rearrangement method to improve the encoding process of the product quantization framework. Our algorithms showed advantages in the form of sensing the distribution of the data, analyzing their degree of aggregation, and rearranging the bit allocation for each subspace, compared to existing algorithms that allocate the bits averagely. The bit rearrangement approach can be integrated with the optimizing strategy proposed in Optimized Product Quantization (OPQ) to produce DS-OPQ. In addition to performing more effectively than OPQ, DS-OPQ converges much faster.

Thus, to solve the mismatch between the data distribution and the bit budget of each subspace, optimization of the data distribution and bit rearrangement based on the data distribution were both shown to be effective. Combination of the two strategies resulted in the most effective way to solve the problem.

In this work, we follow the assumption in OPQ that the optimal solution for data projection is a linear function of the dimensions. However, this may fail to solve some practical problems. Thus, nonlinear projection of the data would be expected to result in more accurate quantization performance. In future, we plan to utilize a kernel function, nonlinear embedding, and deep learning to model optimal data projection.

Appendix

In this Appendix, we provide a formal proof of Lemma 1.

Proof: We define a function to measure the change in the distortions.

$$F = D' - D. \tag{15}$$

Then, we define ε as $\varepsilon = V_1/V_2$, the lemma can be reformulated as, "we have $F < 0$, once $\varepsilon > \varepsilon_*$".

First, F is a monotonic function about ε, where ε is calculated by $\varepsilon = V_1/V_2$. Once the value of ε increases from ε_1 to ε_2 ($\varepsilon_2 > \varepsilon_1$), the differences of V_1 and V_2 will be more obvious. Without loss of generality, we assume that $\mathbf{X}_2$ in the two cases (ε_1 , ε_2) are the same. Then, $V_{1,\varepsilon_2} > V_{1,\varepsilon_1}$. Accordingly, the quantization task of $\mathbf{X}_{1,\varepsilon_2}$ is easier than that of $\mathbf{X}_{1,\varepsilon_1}$ or the allocated bits of $\mathbf{X}_{1,\varepsilon_2}$ is more than that of $\mathbf{X}_{1,\varepsilon_1}$. Then, the representative abilities of the bits in $\mathbf{X}_{1,\varepsilon_2}$ is more redundant than that in $\mathbf{X}_{1,\varepsilon_1}$. Losing one bit will bring less influence to the quantization performance of $\mathbf{X}_{1,\varepsilon_2}$ than to that of $\mathbf{X}_{1,\varepsilon_1}$. Hence, the increment of the distortion of $\mathbf{X}_{1,\varepsilon_2}$ is less obvious than that of $\mathbf{X}_{1,\varepsilon_1}$:

$$D'_{1,\varepsilon_2} - D_{1,\varepsilon_2} < D'_{1,\varepsilon_1} - D_{1,\varepsilon_1}. \tag{16}$$

However, in the two cases (ε_1 and ε_2), the decrement of the distortion of $\mathbf{X}_2$ keeps the same. Eventually, we have

$$D'_{\varepsilon_2} - D_{\varepsilon_2} < D'_{\varepsilon_1} - D_{\varepsilon_1}, \tag{17}$$

or equivalently, $F_{\varepsilon_2} < F_{\varepsilon_1}$. Thus, F is a decreasing function about ε.

Second, when ε approaches ∞, we have $F < 0$. This is the case when V_1 is sufficiently large. Then the data in the first subspace are around one point and the distortion ($D_{1,\infty}$) will not change much even after one bit is donated to another subspace, i.e., $D'_{1,\infty} = D_{1,\infty}$. However, the distortion of $\mathbf{X}_2$ ($D_{2,\infty}$) definitely decreases after receiving one bit, i.e., $D'_{2,\infty} < D_{2,\infty}$. Lastly, the overall distortion decreases ($D'_\infty < D_\infty$) and $F < 0$. Vice versa, when ε approaches 0, $F > 0$.

Eventually, based on the Intermediate Value Theorem [41], we conclude that there must exist a constant ε_* such that, for an arbitrary constant ε, we have $F < 0$ whenever we have $\varepsilon > \varepsilon_*$. This is the same as the statement in the lemma. □

References

[1] O. Boiman, E. Shechtman, and M. Irani, "In defense of nearest-neighbor based image classification," in *Proc. IEEE Conf. CVPR*, Jun. 2008, pp. 1–8.

[2] B. Han, X. Zhao, D. Tao, X. Li, Z. Hu, and H. Hu, "Dayside aurora classification via BIFs-based sparse representation using manifold learning," *Int. J. Comput. Math.*, vol. 91, no. 11, pp. 2415–2426, 2014.

[3] J. Chao, R. Huitl, K. Steinbach, and D. Schroeder, "A novel rate control framework for SIFT/SURF feature preservation in H.264/AVC video compression," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 25, no. 6, pp. 958–972, Nov. 2015.

[4] J. Sivic and A. Zisserman, "Video Google: A text retrieval approach to object matching in videos," in *Proc. 9th IEEE ICCV*, Oct. 2003, pp. 1470–1477.

[5] X. Wu and K. Kashino, "Second-order configuration of local features for geometrically stable image matching and retrieval," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 25, no. 8, pp. 1395–1408, Aug. 2015.

[6] A. Torralba, R. Fergus, and W. T. Freeman, "80 million tiny images: A large data set for nonparametric object and scene recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 30, no. 11, pp. 1958–1970, Nov. 2008.

[7] W. Zhou, M. Yang, X. Wang, H. Li, Y. Lin, and Q. Tian, "Scalable feature matching by dual cascaded scalar quantization for image retrieval," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 38, no. 1, pp. 159–171, Jan. 2016.

[8] J. Wang, J. Wang, N. Yu, and S. Li, "Order preserving hashing for approximate nearest neighbor search," in *Proc. ACM Conf. Multimedia*, New York, NY, USA, Oct. 2013, pp. 133–142.

[9] D. Zhang, G. Yang, Z. Lin, D. Cai, and X. He, "A unified approximate nearest neighbor search scheme by combining data structure and hashing," in *Proc. AAAI Nat. Conf. Artif. Intell.*, Bellevue, WA, USA, Jul. 2013, pp. 681–687.

[10] P. Indyk and R. Motwani, "Approximate nearest neighbors: Towards removing the curse of dimensionality," in *Proc. ACM Symp. Theory Comput. (STOC)*, New York, NY, USA, 1998, pp. 604–613.

[11] Y. Weiss, A. Torralba, and R. Fergus, "Spectral hashing," in *Proc. Adv. Neural Inf. Process. Syst.*, Whistler, BC, Canada, Dec. 2009, pp. 1753–1760.

[12] M. Kan, D. Xu, S. Shan, and X. Chen, "Semisupervised hashing via kernel hyperplane learning for scalable image search," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 24, no. 4, pp. 704–713, Apr. 2014.

[13] B. Kulis and T. Darrell, "Learning to hash with binary reconstructive embeddings," in *Proc. Adv. Neural Inf. Process. Syst.*, Whistler, BC, Canada, Dec. 2009, pp. 1042–1050.

[14] M. Norouzi and D. M. Blei, “Minimal loss hashing for compact binary codes,” in *Proc. Int. Conf. Mach. Learn.*, Bellevue, WA, USA, Jun. 2011, pp. 353–360.

[15] W. Liu, J. Wang, R. Ji, Y.-G. Jiang, and S.-F. Chang, “Supervised hashing with kernels,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Providence, RI, USA, Jun. 2012, pp. 2074–2081.

[16] Y. Weiss, R. Fergus, and A. Torralba, “Multidimensional spectral hashing,” in *Proc. Eur. Conf. Comput. Vis.*, Florence, Italy, Oct. 2012, pp. 340–353.

[17] R. M. Gray and D. L. Neuhoff, “Quantization,” *IEEE Trans. Inf. Theory*, vol. 44, no. 6, pp. 2325–2383, Oct. 1998.

[18] G. J. Sullivan, J.-R. Ohm, W.-J. Han, and T. Wiegand, “Overview of the high efficiency video coding (HEVC) standard,” *IEEE Trans. Circuits Syst. Video Technol.*, vol. 22, no. 12, pp. 1649–1668, Dec. 2012.

[19] H. Jegou, M. Douze, and C. Schmid, “Product quantization for nearest neighbor search,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 1, pp. 117–128, Jan. 2011.

[20] R. Ji *et al.*, “Location discriminative vocabulary coding for mobile landmark search,” *Int. J. Comput. Vis.*, vol. 96, no. 3, pp. 290–314, Feb. 2012.

[21] A. Babenko and V. Lempitsky, “The inverted multi-index,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2012, pp. 3069–3076.

[22] J. Brandt, “Transform coding for fast approximate nearest neighbor search in high dimensions,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, San Francisco, CA, USA, Jun. 2010, pp. 1815–1822.

[23] Y. Gong, S. Lazebnik, A. Gordo, and F. Perronnin, “Iterative quantization: A procrustean approach to learning binary codes for large-scale image retrieval,” *IEEE Trans. Pattern Anal. Mach. Intell*, vol. 35, no. 12, pp. 2916–2929, Dec. 2013.

[24] T. Ge, K. He, Q. Ke, and J. Sun, “Optimized product quantization for approximate nearest neighbor search,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Portland, Oregon, USA, Jun. 2013, pp. 2946–2953.

[25] J.-P. Heo, Z. Lin, and S.-E. Yoon, “Distance encoded product quantization,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Columbus, OH, USA, Jun. 2014, pp. 2139–2146.

[26] H. Hu, “Enhanced Gabor feature based classification using a regularized locally tensor discriminant model for multiview gait recognition,” *IEEE Trans. Circuits Syst. Video Technol.*, vol. 23, no. 7, pp. 1274–1286, Jul. 2013.

[27] H. Cheng, Z. Liu, L. Hou, and J. Yang, “Sparsity-induced similarity measure and its applications,” *IEEE Trans. Circuits Syst. Video Technol.*, vol. 26, no. 4, pp. 613–626, Apr. 2016.

[28] Z. Wang, J. Feng, S. Yan, and H. Xi, “Linear distance coding for image classification,” *IEEE Trans. Image Process.*, vol. 22, no. 2, pp. 537–548, Feb. 2013.

[29] X. Sun, C. Wang, C. Xu, and L. Zhang, “Indexing billions of images for sketch-based retrieval,” in *Proc. ACM Conf. Multimedia*, New York, NY, USA, Oct. 2013, pp. 233–242.

[30] M. Rusiñol, D. Aldavert, R. Toledo, and J. Lladós, “Efficient segmentation-free keyword spotting in historical document collections,” *Pattern Recognit.*, vol. 48, no. 2, pp. 545–555, Feb. 2015.

[31] Y.-G. Jiang, Y. Jiang, and J. Wang, “VCDB: A large-scale database for partial copy detection in videos,” in *Proc. Eur. Conf. Comput. Vis.*, Zurich, Switzerland, Sep. 2014, pp. 357–371.

[32] R. Ji, H. Yao, W. Liu, X. Sun, and Q. Tian, “Task-dependent visual-codebook compression,” *IEEE Trans. Image Process.*, vol. 21, no. 4, pp. 2282–2293, Apr. 2012.

[33] Y. Cho, D.-K. Kwon, J. Liu, and C.-C. J. Kuo, “Dependent R/D modeling techniques and joint T-Q layer bit allocation for H.264/SVC,” *IEEE Trans. Circuits Syst. Video Technol.*, vol. 23, no. 6, pp. 1003–1015, Feb. 2013.

[34] J. Revaud, M. Douze, C. Schmid, and H. Jégou, “Event retrieval in large video collections with circulant temporal encoding,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Portland, OR, USA, Jun. 2013, pp. 2459–2466.

[35] N. Inoue and K. Shinoda, “Neighbor-to-neighbor search for fast coding of feature vectors,” in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2013, pp. 1233–1240.

[36] M. Norouzi and D. J. Fleet, “Cartesian k-means,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Portland, OR, USA, Jun. 2013, pp. 3017–3024.

[37] Y. Kalantidis and Y. Avrithis, “Locally optimized product quantization for approximate nearest neighbor search,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Columbus, OH, USA Jun. 2014, pp. 2329–2336.

[38] T. Ge, K. He, Q. Ke, and J. Sun, “Optimized product quantization,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 36, no. 4, pp. 744–755, Dec. 2014.

[39] T. Zhang, C. Du, and J. Wang, “Composite quantization for approximate nearest neighbor search,” in *Proc. IEEE Int. Conf. Mach. Learn.*, Peking, China, Jun. 2014, pp. 838–846.

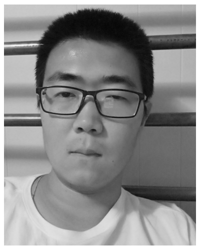
[40] X. Liu, B. Du, C. Deng, M. Liu, and B. Lang, “Structure sensitive hashing with adaptive product quantization,” *IEEE Trans. Cybern.*, vol. 46, no. 10, pp. 2252–2264, Oct. 2016.

[41] T. M. Apostol, *Calculus*. Hoboken, NJ, USA: Wiley, 1967.

[42] M. Douze, H. Jégou, and F. Perronnin, “Polysemous codes,” in *Proc. Eur. Conf. Comput. Vis.*, Amsterdam, The Netherlands, Oct. 2016, pp. 785–801.

[43] G. Huang, M. Ramesh, T. Berg, and E. Learned-Miller, “Labeled faces in the wild: A database for studying face recognition in unconstrained environments,” Univ. Massachusetts, Amherst, Amherst, MA, USA, Tech. Rep. 07-49, Oct. 2008.

[44] L. Li, W. Huang, I. Y.-H. Gu, and Q. Tian, “Statistical modeling of complex backgrounds for foreground object detection,” *IEEE Trans Image Process.*, vol. 13, no. 11, pp. 1459–1472, Nov. 2004.



Linhao Li was born in 1989. He received the B.S. degree in applied mathematics and the M.S. degree in computational mathematics from Tianjin University in 2012 and 2014, respectively, where he is currently pursuing the Ph.D. degree with the School of Computer Science and Technology. His research interests focus on quantization and hashing learning, sparse signal recovery, background modeling, and foreground detection.

Qinghua Hu received the B.S., M.S., and Ph.D. degrees from the Harbin Institute of Technology, Harbin, China, in 1999, 2002, and 2008, respectively. He was a Post-Doctoral Fellow with the Department of Computing, Hong Kong Polytechnic University, from 2009 to 2011. He is currently a Full Professor and the Vice Dean of the School of Computer Science and Technology, Tianjin University. His research interests are focused on rough sets, granular computing, and data mining for classification and regression. He has published more than 100 journal and conference papers in the areas of granular computing-based machine learning, reasoning with uncertainty, pattern recognition, and fault diagnosis. He was the PC Co-Chair of RSCTC 2010, CRSSC 2012, 2014, RSKT 2014, and ICMLC 2014, and serves as a referee for a great number of journals and conferences.

Yahong Han received the Ph.D. degree from Zhejiang University, Hangzhou, China, in 2012. From 2014 to 2015, he was a Visiting Scholar with the Prof. B. Yus Group, University of California at Berkeley, Berkeley, CA, USA. He is currently an Associate Professor with the School of Computer Science and Technology, Tianjin University, Tianjin, China. His current research interests include multimedia analysis, retrieval, and machine learning.

Xin Li received the B.S. degree (Hons.) in electronic engineering and information science from the University of Science and Technology of China, Hefei, China, and the Ph.D. degree in electrical engineering from Princeton University, Princeton, NJ, USA, in 1996 and 2000, respectively.

He was a member of Technical Staff with Sharp Laboratories of America, Camas, WA, USA, from 2000 to 2002. Since 2003, he has been a Faculty Member with the Lane Department of Computer Science and Electrical Engineering, West Virginia University. His current research interests include image and video coding and processing. He is currently a member of the Image, Video, and Multidimensional Signal Processing Technical Committee and an Associate Editor of the IEEE TRANSACTIONS ON IMAGE PROCESSING.