Optimized Product Quantization

Tiezheng Ge, Kaiming He†, Qifa Ke, and Jian Sun

Abstract—Product quantization (PQ) is an effective vector quantization method. A product quantizer can generate an exponentially large codebook at very low memory/time cost. The essence of PQ is to decompose the high-dimensional vector space into the Cartesian product of subspaces and then quantize these subspaces separately. The optimal space decomposition is important for the PQ performance, but still remains an unaddressed issue. In this paper, we optimize PQ by minimizing quantization distortions w.r.t. the space decomposition and the quantization codebooks. We present two novel solutions to this challenging optimization problem. The first solution iteratively solves two simpler sub-problems. The second solution is based on a Gaussian assumption and provides theoretical analysis of the optimality. We evaluate our optimized product quantizers in three applications: (i) compact encoding for exhaustive ranking [1], (ii) building inverted multi-indexing for non-exhaustive search [2], and (iii) compacting image representations for image retrieval [3]. In all applications our optimized product quantizers outperform existing solutions.

Index Terms—Vector quantization, nearest neighbor search, image retrieval, compact encoding, inverted indexing

1 Introduction

Approximate nearest neighbor (ANN) search is of great importance in many computer vision problems, such as image/video retrieval [4], image classification [5], and object/scene recognition [6]. Vector Quantization (VQ) [7] is a popular and successful method for ANN search. The vector quantization method is used in two ways for ANN: (i) to build inverted indexing [4] for non-exhaustive search, or (ii) to encode vectors into compact codes [1], [8], [9] for exhaustive search. In the non-exhaustive search, the quantizers can be k-means [4] and its variants [10]. A query is quantized into a codeword and then compared with a short list of data which have the same or similar codewords. In the exhaustive search, the data are quantized into codewords [1], [8], [9]; the distances of vectors are approximated by the distances of codewords. With a few dozens of bits used per vector, the memory footprint is small, and the search speed is fast. The compact encoding methods can be combined with inverted indexing (*e.g.*, as in [1], [2]) to achieve real-time and high-quality search in billions of vectors.

A *product quantizer* [7] is a solution to VQ when an exponentially large number of codewords are desired. The key idea is to decompose the original vector space into the Cartesian product of M low-dimensional subspaces and quantize each subspace into k codewords. The effective number of codewords in the original space is k^M, but the cost of storing them is merely $O(Dk)$ for D-dimensional vectors.

Such a Product Quantization (PQ) technique has been used for compact encoding for approximate distance computation in [1], and recently has also been adopted for building inverted indexing in [2]. When used for compact encoding, the time complexity is $O(Mk)$ per distance computation using look-up tables [1]. When used for building inverted indexing [2], a query can quickly find its nearest codewords (out of k^M codewords) in $O(Mk)$ time. Currently, [1] is among the state-of-the-art compact encoding methods, whereas [2] is among the state-of-the-art inverted indexing methods.

Despite the power of PQ, the optimal decomposition of the vector space remains a largely unaddressed issue. In [1] it has been noticed that the prior knowledge about the structures of the input data (SIFT/GIST) is of particular importance, and the search accuracy would become substantially worse if such knowledge is not available. But the strong reliance on the prior knowledge largely limits the performance of PQ on general data, *e.g.*, raw image pixels, compressed representations (by PCA, sparse coding, *etc.*), and mixed representations. The usage of the prior structures also constrains the choice of the subspace number M and the codebook size. In case of no structure, [3] propose to optimize a Householder transform such that the vectors have balanced variances in all components. In [3] it is also observed that a random rotation performs similarly to the Householder transform. But the optimality in terms of quantization error remains unclear in both cases.

In this paper, we formulate PQ as an optimization problem that minimizes the quantization distortion by seeking for optimal codewords and space decom-

† *Correspondence author.*

- *T. Ge is with the University of Science and Technology of China, Hefei, China. E-mail: getzh@mail.ustc.edu.cn*
- *K. He and J. Sun are with the Visual Computing Group, Microsoft Research Asia, Beijing, China. E-mail: {kahe,jiansun}@microsoft.com*
- *Q. Ke is with the Microsoft Bing, Sunnyvale, CA, US. E-mail: qke@microsoft.com*

positions. Such an optimization problem is challenging due to the large number of unknown parameters. In this work we present two solutions. In the first solution, we iteratively solve two simpler subproblems: solving for the space decomposition with the codewords fixed, and vice versa. This solution is non-parametric in that it does not assume any priori information about the data distribution. Our second solution is a parametric one in that it assumes the data follows a Gaussian distribution. Under this assumption, we derive an analytical formulation of the lower bound of the quantization distortion. Then we theoretically prove that this lower bound is minimized when (i) the subspaces are mutually *independent*, and simultaneously (ii) the vectors have *balanced* variances in all subspaces. Based on these theories, we propose a simple *Eigenvalue Allocation* method to effectively optimize the space decomposition.

We demonstrate by experiments the superiority of our methods in three applications:

- When used for compact encoding for exhaustive search, our method outperforms several variants of PQ and other VQ methods. Our method is also substantially better than various state-of-the-art binarization methods.
- When used for building inverted indexing, our method improves the "inverted multi-index" [2] by optimizing its codebook. The performance can be improved even further when our optimized PQ is adopted in the combination of inverted multi-index and compact encoding. This marks the state-of-the-art performance on the one billion SIFT dataset.
- We further apply our optimized PQ to compress image representations (VLAD [3] and Fisher Vectors [11]) for image retrieval. Our method provides better retrieval accuracies than the original PQ encoding method.

A preliminary version of this work was published in CVPR 2013 [12]. Concurrent with [12], a very similar work "Cartesian k-means" [13] is independently developed. We note that our first solution (Sec. 3.1) is equivalent to the Cartesian k-means method. But our second solution (Sec. 3.2) takes another step. It provides theoretical guarantees of the optimality under some practical assumption. It also provides a way of initialization (better than a random one) for the non-parametric solution.

We have published the Matlab code[1] of our two solutions for the ease of practical applications and theoretical analysis.

2 Quantization Distortion

A variety of ANN methods, including k-means [14], Product Quantization [1], and Iterative Quantization [9], can be formulated within a framework of vector quantization [7]. The quantization distortion is a common objective function among the different methods studied. We can treat the specific configuration of each method as the *constraints* when optimizing the common objective function, rather than incorporate the configuration into the objective function. In this way, various methods (including the proposed) can be discussed in a unified framework.

2.1 Vector Quantization

A vector quantization (VQ) system [7] maps a vector $\mathbf{x} \in \mathbb{R}^D$ to a *codeword* $\mathbf{c}$ in a *codebook* $\mathcal{C} = \{\mathbf{c}(i)\}$ with i in a finite index set. The mapping, termed as a *quantizer*, is denoted by: $\mathbf{x} \rightarrow \mathbf{c}(i(\mathbf{x}))$. The function $i(\cdot)$ is called an *encoder*, and function $\mathbf{c}(\cdot)$ is called a *decoder* [7][2].

The *quantization distortion* E is defined as:

$$E = \frac{1}{n} \sum_{\mathbf{x}} \|\mathbf{x} - \mathbf{c}(i(\mathbf{x}))\|^2, \tag{1}$$

where $\|\cdot\|$ is the Euclidean distance, and n is the total number of data samples. Note this definition of distortion applies to any quantizer, no matter what the encoder and decoder are.

Given a codebook $\mathcal{C}$, an encoder that minimizes the distortion E must satisfy the first Lloyd's condition [7]: the encoder $i(\mathbf{x})$ should map any $\mathbf{x}$ to its nearest codeword in the codebook $\mathcal{C}$. Also note this property is valid no matter what the codebook is. Such a quantizer is known as a *Voronoi* quantizer [7].

2.2 Codebook Generation

Minimizing the distortion under different constraints leads to different methods.

2.2.1 K-means

If there is no constraint on the codebook, minimizing the distortion in (1) leads to the classical k-means [14]. With the encoder $i(\cdot)$ fixed, a codeword $\mathbf{c}(i)$ is the mean of the vectors that are indexed as i – this is the second Lloyd's condition [7].

2.2.2 Product Quantization

If any codeword $\mathbf{c}$ must be taken from the Cartesian product of sub-codebooks, minimizing the distortion in (1) leads to the PQ method [1].

Formally, denote any $\mathbf{x} \in \mathbb{R}^D$ as the concatenation of M subvectors: $\mathbf{x} = [\mathbf{x}^1, ...\mathbf{x}^m, ...\mathbf{x}^M]$. For simplicity we assume the subvectors have an equal number of dimensions D/M. The Cartesian product $\mathcal{C} = \mathcal{C}^1 \times ... \times$

1. research.microsoft.com/en-us/um/people/kahe/

2. In this paper, we abuse the notations of $i(\cdot)$ and i to simplify the presentation: by $i(\cdot)$ we mean the mapping of a vector to an index, and by i we mean the index of a certain codeword. Likewise, we abuse the notations of $\mathbf{c}(\cdot)$ and $\mathbf{c}$: by $\mathbf{c}(\cdot)$ we mean the mapping of an index to a codeword, and by $\mathbf{c}$ we mean a certain codeword.

$\mathcal{C}^M$ is the codebook in which any codeword $\mathbf{c} \in \mathcal{C}$ concatenates M sub-codewords: $\mathbf{c} = [\mathbf{c}^1, ...\mathbf{c}^m, ...\mathbf{c}^M]$, with each $\mathbf{c}^m \in \mathcal{C}^m$. The objective function of PQ, though not explicitly defined in [1], is essentially:

$$\min_{\mathcal{C}^1,...,\mathcal{C}^M} \sum_{\mathbf{x}} \|\mathbf{x} - \mathbf{c}(i(\mathbf{x}))\|^2, \quad (2)$$
$$s.t. \quad \mathbf{c} \in \mathcal{C} = \mathcal{C}^1 \times ... \times \mathcal{C}^M.$$

It is easy to show that $\mathbf{x}$'s nearest codeword $\mathbf{c}$ in $\mathcal{C}$ is the concatenation of the M nearest sub-codewords $\mathbf{c} = [\mathbf{c}^1, ...\mathbf{c}^m, ...\mathbf{c}^M]$ where $\mathbf{c}^m$ is the nearest sub-codeword of the subvector $\mathbf{x}^m$. So (2) can be split into M separate subproblems, each of which can be solved by k-means. This is exactly the way of PQ [1].

PQ can easily generate a codebook $\mathcal{C}$ with an exponentially large number of codewords. If each sub-codebook has k sub-codewords, then their Cartesian product $\mathcal{C}$ has k^M codewords. The cost of storing these codewords is merely $O(Mk{\cdot}D/M) = O(Dk)$. This is not possible for classical k-means when k^M is large.

The PQ developed in [1] is used for compact encoding and approximate distance computation. The cost of storing each encoded vector is $M \log_2 k$ bits. The distance between a query and any vector is approximated by the distance of their codewords (known as *Symmetric Distance Computation* or SDC), or by the distance between the query and the codeword of the vector (known as *Asymmetric Distance Computation* or ADC). Both ways of distance computation are efficient using lookup tables. For SDC, the distances between any two sub-codewords in a subspace are pre-computed and stored in a k-by-k lookup table. For ADC, the distances between the sub-vector of the query and the sub-codewords in a subspace are pre-computed on-line and stored in a 1-by-k lookup table. The distance in the original space is simply the sum of the distances computed from the M subspaces.

2.2.3 Iterative Quantization

If any codeword $\mathbf{c}$ must be taken from "the vertexes of a rotating hyper-cube", minimizing the distortion leads to a binary embedding method called Iterative Quantization (ITQ) [9].

The D-dimensional vectors in $\{-a, a\}^D$ are the vertices of an axis-aligned D-dimensional hyper-cube. Suppose the data has been zero-centered. The objective function in ITQ [9] is essentially:

$$\min_{R,a} \sum_{\mathbf{x}} \|\mathbf{x} - \mathbf{c}(i(\mathbf{x}))\|^2, \quad (3)$$
$$s.t. \quad \mathbf{c} \in \mathcal{C} = \{\mathbf{c} \mid R\mathbf{c} \in \{-a, a\}^D\}, \quad R^{\mathrm{T}}R = I,$$

where R is an orthogonal matrix and I is an identity matrix. The codebook $\mathcal{C}$ contains 2^D codewords.

This optimization problem is iteratively solved in [9] just like k-means: update the codebook (represented by R and a) with the index fixed, and vise versa. In [9] it has also shown that the length a in (3) does not impact the resulting partition planes. We keep a here because it matters when we compare the distortion with other quantization methods.

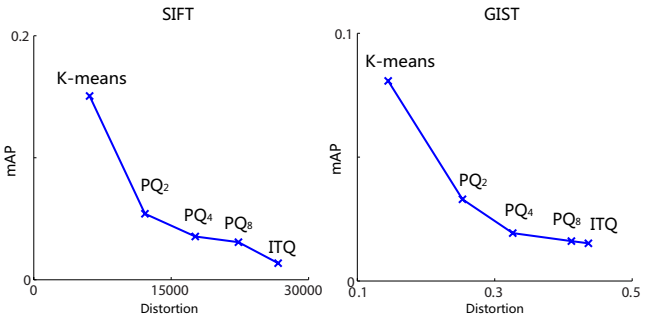


Fig. 1: mAP *vs.* quantization distortion.

With a rotating hyper-cube, the squared Euclidean distance between any two codewords is equivalent to the Hamming distance. So ITQ can be viewed in the category of binary hashing methods [15], [16], [17], [18], [19], [20]. The formulation in (3) also indicates that any orthogonal binary hashing method can be viewed a vector quantizer.

2.3 Distortion as the Objective Function

The above methods all optimize the same form of quantization distortion, but subject to different constraints. This implies that distortion is an objective function that can be evaluated across different quantization methods. The common usage of distortion as objective functions also implies that it generally impacts the ANN search accuracy. We empirically verify that the distortion is tightly correlated to the ANN search accuracy of different methods.

To show this, we investigate the ANN search accuracy on two large datasets (SIFT1M and GIST1M [1]). The data vectors are ranked by SDC (for ITQ this is equivalent to Hamming ranking). Then we compute the mean Average Precision (mAP) over all queries. We consider the 100 nearest neighbors as the ground truth.

In this test we use $B = 16$ bits for each codeword. This essentially gives $K = 2^{16}$ codewords. Here we do not use larger B because we want to involve k-means (since $K = 2^B$). We test five quantization methods: k-means, ITQ, and three variants of PQ (decomposed into $M = 2$, 4, or 8 subspaces, denoted as PQ_M). The mAP *vs.* the distortion are shown in Fig. 1. We can see that the mAP has a strong relation to the quantization distortion. We also find this relation is valid under various ANN metrics besides mAP (like precision/recall at the top N ranked data), with various number (1 to 10^4) of ground truth nearest neighbors.

The PQ paper [1] has given some statistical relationship between distortion and distance estimation (therefore related to ANN search). Our experiment

shows this relationship remains valid if k-means and ITQ are also considered.

In Fig. 1 we also see stricter constraints lead to higher distortion. For example, the solution space of PQ_4 is a subset of the solution space of PQ_2 (if $\mathbf{c}$ is in $\mathcal{C}^1 \times ... \times \mathcal{C}^4$ for some $\frac{D}{4}$-dimensional sub-codebooks $\mathcal{C}^1, ..., \mathcal{C}^4$, then $\mathbf{c}$ must be in $\mathcal{C'}^1 \times \mathcal{C'}^2$ for some $\frac{D}{2}$-dimensional sub-codebooks $\mathcal{C'}^1$, $\mathcal{C'}^2$; but not vice versa). As a result, the distortion of PQ_4 must be no less than the distortion of PQ_2. To the extreme when $M = 1$ and $k = K$ (equivalent to k-means), the distortion is the smallest because no constraint is imposed. However, when k is getting larger, it is memory/time-consuming to maintain the k-by-k lookup tables for distance computation. In practice, k is often kept as the largest affordable number (256 in [1]), and $M = B/\log_2 k$ is then fixed given the pre-defined code length B.

3 Optimized Product Quantization

We use the quantization distortion as an objective function to evaluate the optimality of a product quantizer. A product quantizer involves decomposing the D-dimensional vector space into M subspaces and computing a sub-codebook for each subspace. The optimization problem in (2) is only w.r.t. the sub-codebooks, but not the space decomposition. In this work we consider an optimization problem w.r.t. both.

We use an orthogonal matrix R to represent the space decomposition. Note that any re-ordering (permutation) of the dimensions can be represented by an orthogonal matrix. So R decides all degrees-of-freedom of decomposing a space into M equal-dimensional subspaces.

Considering the sub-codebooks and the space decomposition, we optimize the following objective:

$$\min_{R,\mathcal{C}^1,...,\mathcal{C}^M} \sum_{\mathbf{x}} \|\mathbf{x} - \mathbf{c}(i(\mathbf{x}))\|^2, \tag{4}$$
$$s.t. \quad \mathbf{c} \in \mathcal{C} = \{\mathbf{c} \mid R\mathbf{c} \in \mathcal{C}^1 \times ... \times \mathcal{C}^M, \quad R^{\mathrm{T}}R = I\}$$

In this problem, the free parameters consist of the sub-codebooks $(\mathcal{C}^1, ..., \mathcal{C}^M)$ and the space decomposition R. The additional free parameters of R allows the vector space to rotate, thus relax the constraints on the codewords. So the optimized product quantizer can reduce the quantization distortion versus a pre-fixed R.

Assigning $\mathbf{x}$ to its nearest codeword $\mathbf{c}$ is equivalent to assigning $R\mathbf{x}$ to the nearest $R\mathbf{c}$. To apply the optimized quantizer for encoding, we only need to pre-process the data by $R\mathbf{x}$, and the remaining steps are the same as those in PQ.

We are not the first to consider a problem like (4). In [3] a similar objective function has been mentioned[3], but was thought as "not tractable" possibly due to the coupling of R and $\mathcal{C}^1, ..., \mathcal{C}^M$. Previous methods pre-process the data using a pre-fixed R based on simple heuristics, like randomly ordering the dimensions [1] or randomly rotating the space [3]. The matrix R has not been considered in any optimization coupling the sub-codebooks.

In the following we propose two solutions to the optimization in (4).

3.1 A Non-Parametric Solution

Our non-parametric solution does not assume any data distribution[4]. We split the problem in (4) into two simpler sub-problems.

Step (i): Fix R and optimize $\{\mathcal{C}^m\}_{m=1}^M$.

Denote $\hat{\mathbf{x}} = R\mathbf{x}$ and $\hat{\mathbf{c}} = R\mathbf{c}$. Since R is orthogonal, we have $\|\mathbf{x} - \mathbf{c}\|^2 = \|\hat{\mathbf{x}} - \hat{\mathbf{c}}\|^2$. With R fixed, (4) then becomes:

$$\min_{\mathcal{C}^1,...,\mathcal{C}^m} \sum_{\hat{\mathbf{x}}} \|\hat{\mathbf{x}} - \hat{\mathbf{c}}(i(\hat{\mathbf{x}}))\|^2, \tag{5}$$
$$s.t. \quad \hat{\mathbf{c}} \in \mathcal{C}^1 \times ... \times \mathcal{C}^M.$$

This is the same problem as PQ in (2). We can separately run k-means in each subspace to compute the sub-codebooks.

Step (ii): Fix $\{\mathcal{C}^m\}_{m=1}^M$ and optimize R.

Since $\|\mathbf{x} - \mathbf{c}\|^2 = \|R\mathbf{x} - \hat{\mathbf{c}}\|^2$, the sub-problem becomes:

$$\min_{R} \sum_{\mathbf{x}} \|R\mathbf{x} - \hat{\mathbf{c}}(i(\hat{\mathbf{x}}))\|^2, \tag{6}$$
$$s.t. \quad R^{\mathrm{T}}R = I.$$

The codeword $\hat{\mathbf{c}}(i(\hat{\mathbf{x}}))$ is fixed in this subproblem. It is the concatenation of the M sub-codewords of the sub-vectors in $\hat{\mathbf{x}}$. We denote $\hat{\mathbf{c}}(i(\hat{\mathbf{x}}))$ as $\mathbf{y}$. Given n training samples, we denote X and Y as two D-by-n matrices whose columns are the samples $\mathbf{x}$ and $\mathbf{y}$ respectively. Then we can rewrite (6) as:

$$\min_{R} \|RX - Y\|_{\mathrm{F}}^2, \tag{7}$$
$$s.t. \quad R^{\mathrm{T}}R = I,$$

where $\|\cdot\|_{\mathrm{F}}$ is the Frobenius norm. This is the Orthogonal Procrustes problem [21]. It has a closed-form solution: first apply Singular Value Decomposition (SVD) to $XY^{\mathrm{T}} = USV^{\mathrm{T}}$, and then let $R = VU^{\mathrm{T}}$. In [9] this solution was used to optimize the ITQ problem in (3).

Our algorithm iteratively optimizes Step (i) and (ii). In Step (i) we need to run k-means, which by itself is an iterative algorithm. But we can refine the results from the previous step instead of restarting k-means.

3. See Eqn.(8) of [3].

4. We follow the terminology in statistics that a "non-parametric" model is the one that does not rely on any assumption about the data distribution, while a "parametric" model explicitly assumes certain parameterized distribution such as Gaussian distribution.

Algorithm 1 Non-Parametric OPQ

Input: training samples $\{\mathbf{x}\}$, number of subspaces M, number of sub-codewords k in each sub-codebook.

Output: the matrix R, sub-codebooks $\{\mathcal{C}^m\}_{m=1}^M$, M sub-indices $\{i^m\}_{m=1}^M$ for each $\mathbf{x}$.

1: Initialize R, $\{\mathcal{C}^m\}_{m=1}^M$, and $\{i^m\}_{m=1}^M$.
2: **repeat**
3: Step(i): project the data: $\hat{\mathbf{x}} = R\mathbf{x}$.
4: **for** $m = 1$ to M **do**
5: for $j = 1$ to k: update $\hat{\mathbf{c}}^m(j)$ by the sample mean of $\{\hat{\mathbf{x}}^m \mid i^m(\hat{\mathbf{x}}^m) = j\}$.
6: for $\forall \hat{\mathbf{x}}^m$: update $i^m(\hat{\mathbf{x}}^m)$ by the sub-index of the sub-codeword $\hat{\mathbf{c}}^m$ that is nearest to $\hat{\mathbf{x}}^m$.
7: **end for**
8: Step(ii): solve R by (7).
9: **until** max iteration number reached

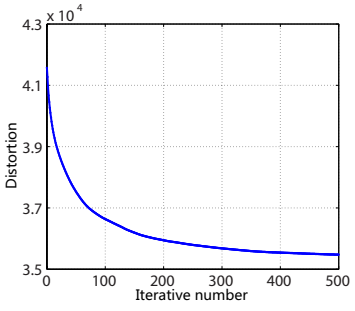


Fig. 2: Convergence of Algorithm 1 in SIFT1M [1]. Here we show $M = 4$ and $k = 256$ (32 bits).

With this strategy, we empirically find that even if only one k-means iteration is run in each Step (i), our entire algorithm still converges to a good solution. A pseudo-code is in Algorithm 1. This algorithm is applicable even when D is not divisible by the subspace number M.

If we ignore line 3 and line 8 in Algorithm 1, it is equivalent to PQ (for PQ one might usually put line 2 in the inner loop). So this algorithm is similar to PQ, except that in each iteration it updates R (line 8) and transforms the data by R (line 3). The SVD of R is $O(D^3)$ complex.

Fig. 2 shows the convergence of our algorithm. In practice we find 100 iterations are good enough for the purpose of ANN search. Like many other alternating optimization algorithms, our algorithm is locally optimal and the final solution depends on the initialization. In the next subsection we propose a parametric solution that can be used to initialize the above iterative algorithm, when no other prior knowledge is given.

3.2 A Parametric Solution

We further propose another solution assuming the data follows a parametric Gaussian distribution. This parametric solution has both practical and theoretical merits. First, it is a simpler method and is optimal if the data follows Gaussian distributions. Second, it provides a way to initialize the non-parametric method. Third, it provides new theoretical explanations for two commonly used criteria in some other ANN methods.

A technical challenge in optimizing (4) is the coupling of R and $\{\mathcal{C}^m\}_{m=1}^M$. We discover that under a Gaussian assumption, the lower bound of the distortion in (4) has an analytical form only depending on R but not $\{\mathcal{C}^m\}_{m=1}^M$. This allows us to directly optimize w.r.t. R.

The Gaussian assumption is only for the purpose of theoretical derivations; the usage of the parametric solution does not rely on this assumption (it only impacts the validity of the optimality). This also happens in the derivations of Spectral Hashing [16].

3.2.1 Distortion Bound of Quantization

We first assume $\mathbf{x} \in \mathbb{R}^D$ is subject to a Gaussian distribution with zero mean: $\mathbf{x} \sim \mathcal{N}(0, \Sigma)$. Here Σ is the D-by-D covariance matrix. From the *rate distortion theory* [22], the distortion E satisfies:

$$E \geq k^{-\frac{2}{D}} D |\Sigma|^{\frac{1}{D}}, \tag{8}$$

where $|\Sigma|$ is the determinant. This inequality gives the distortion lower bound for any quantizer with k codewords. The following table shows the values of this bound and the empirical distortion of a k-means quantizer (10^5 samples, $k = 256$, σ_d^2 randomly generated in $[0.5, 1]$):

D	32	64	128
bound	16.2	38.8	86.7
empirical E	17.1±0.035%	39.9±0.025%	88.5±0.021%

TABLE 1

This table implies that it is reasonable to consider the bound in (8) as an approximation to the k-means distortion. The small gap ($\sim 5\%$) may be due to two reasons: a k-means quantizer can only achieve a locally optimal solution, and the fixed code-length for all codewords may not achieve optimal bit rate (in information theory, it is possible to reduce the average bit rate by varying the bit-length of codewords, known as *entropy encoding* [22]).

3.2.2 Distortion Bound of Product Quantization

Next we study the distortion bound of a product quantizer. We still assume $\mathbf{x} \sim \mathcal{N}(0, \Sigma)$. When applying R to data, the variable $\hat{\mathbf{x}} = R\mathbf{x}$ is subject to another Gaussian distribution: $\hat{\mathbf{x}} \sim \mathcal{N}(0, \hat{\Sigma})$ with $\hat{\Sigma} = R\Sigma R^{\mathrm{T}}$.

We can decompose $\hat{\Sigma}$ into $M \times M$ sub-matrices:

$$\hat{\Sigma} = \begin{pmatrix} \hat{\Sigma}_{11} & \cdots & \hat{\Sigma}_{1M} \\ \vdots & \ddots & \vdots \\ \hat{\Sigma}_{M1} & \cdots & \hat{\Sigma}_{MM} \end{pmatrix}. \quad (9)$$

Here the diagonal sub-matrices $\hat{\Sigma}_{mm}$ are the covariance of the m-th subspace. Notice $\hat{\mathbf{x}}^m$ subjects to $\frac{D}{M}$-dimensional Gaussian $\mathcal{N}(0, \hat{\Sigma}_{mm})$, so from (8), the distortion of the m-th subspace is no smaller than $k^{-\frac{2M}{D}} \frac{D}{M} |\hat{\Sigma}_{mm}|^{\frac{M}{D}}$. So the distortion of PQ satisfies:

$$E(R) \geq k^{-\frac{2M}{D}} \frac{D}{M} \sum_{m=1}^{M} |\hat{\Sigma}_{mm}|^{\frac{M}{D}}. \quad (10)$$

This gives us an analytical form of the lower bound of the distortion. This lower bound does not depends on the codewords.

3.2.3 Minimizing the Distortion Bound

If the lower bound is reasonably tight (see Table 1), we can expect reasonably minimized distortion through minimizing its lower bound. So we propose to minimize the distortion bound in (10) w.r.t. R :

$$\min_{R} \sum_{m=1}^{M} |\hat{\Sigma}_{mm}|^{\frac{M}{D}}, \quad (11)$$
$$s.t. \quad R^{\mathrm{T}} R = I,$$

The constant scale in (10) has been ignored. This problem belongs to the category of "optimizing a function with orthogonal constraints" [23]. Due to the orthogonal constraint, such a problem is non-convex in general [23]. An iterative algorithm has been developed in [23], but its quality is still sensitive to the initialization. Fortunately, the special form of our objective function can be minimized by a simple algorithm, as we show next.

We find the objective in (11) has a constant lower bound independent of R. This lower bound is achievable under a very mild assumption. As a result, optimizing the objective in (11) is equivalent to achieving its lower bound.

Using the *inequality of arithmetic and geometric means* (AM-GM) [24], the objective in (11) satisfies:

$$\sum_{m=1}^{M} |\hat{\Sigma}_{mm}|^{\frac{M}{D}} \geq M \prod_{m=1}^{M} |\hat{\Sigma}_{mm}|^{\frac{1}{D}}. \quad (12)$$

The equality is achieved if and only if the term $|\hat{\Sigma}_{mm}|$ has the same value for all m.

Further, in matrix analysis [25] the *Fischer's inequality* gives:

$$\prod_{m=1}^{M} |\hat{\Sigma}_{mm}| \geq |\hat{\Sigma}|. \quad (13)$$

The equality is achieved if and only if the off-diagonal sub-matrices in $\hat{\Sigma}$ equal to a zero matrix. Here $|\hat{\Sigma}| \equiv |\Sigma|$ is a constant independent of R.

Combining (12) and (13), we obtain the lower bound for the distortion bound:

$$\sum_{m=1}^{M} |\hat{\Sigma}_{mm}|^{\frac{M}{D}} \geq M |\Sigma|^{\frac{1}{D}}. \quad (14)$$

The lower bound is achieved if the achievability in (12) and (13) are both satisfied:

(i) Independence. If we align the data by PCA, the equality in Fischer's inequality (13) is achieved. This implies we make the subspaces independent to each other.

(ii) Balanced the Variances of Subspaces. The equality in AM-GM (12) is achieved if $|\hat{\Sigma}_{mm}|$ has the same value for all subspaces. Suppose the data has been aligned by PCA. Then $|\hat{\Sigma}_{mm}|$ equals to the product of the eigenvalues of Σ_{mm}. We make a mild assumption that by re-ordering the principal components, we can balance the product of eigenvalues for each subspace (so the values $|\hat{\Sigma}_{mm}|$ are equal). As a result, both equalities in AM-GM (12) and Fischer's (13) are satisfied, so the objective function in (12) is minimized.

3.2.4 Algorithm: Eigenvalue Allocation

Based on the above analysis, we propose a simple *Eigenvalue Allocation* method to optimize (11). This method is a greedy solution to the combinatorial "balanced partition" problem [26].

We first align the data using PCA and sort the eigenvalues σ^2 in the descending order $\sigma_1^2 \geq ... \geq \sigma_D^2$. It is not necessary to reduce dimensions. We prepare M empty buckets, each for one of the M subspaces. We sequentially pick out the largest eigenvalue and allocate it to the bucket having the minimum product of the eigenvalues in it (unless this bucket is full, *i.e.*, with D/M eigenvalues in it). The eigenvalues in each bucket provide the principal components (eigenvectors) that will be used to form each subspace. In fact, this algorithm re-orders the eigenvectors to form the columns of R.

In real data sets, we find this greedy algorithm is sufficiently good for minimizing the objective function in (11). To show this fact, we compute the covariance matrix Σ from the SIFT1M/GIST1M datasets. The following table shows the lower bound of the objective function (right hand side in (14)) and the objective function value (left hand side in (14)) optimized by our Eigenvalue Allocation algorithm. Here we use $M = 8$ and $k = 256$.

	theoretical min	Eigen Allocation
SIFT	2.9286×10^3	2.9287×10^3
GIST	1.9870×10^{-3}	1.9870×10^{-3}

TABLE 2

We can see the above greedy algorithm well achieves the theoretical lower bound.

Summary of the parametric solution. Our parametric solution first computes the $D \times D$ covariance matrix Σ of the data and uses Eigenvalue Allocation to generate R. The data are then transformed by R. The PQ algorithm is then performed on the transformed data.

The derivation of this solution requires D to be divisible by M due to (10). But in practice, the usage of the Eigenvalue Allocation does not need this assumption.

3.2.5 Discussion

Interestingly, some existing methods have adopted the criteria of "independence" or "balance", either heuristically or in objective functions different from ours.

Under the Gaussian assumption, in [16], [27], [8] the "independence" criterion is done by de-correlation via PCA. These methods are derived from other objective functions different from ours.

The "balance" criterion was used in [3], [16], [8]. The method in [3] rotates the data by a Householder transform to balance the variances of all *components* (*dimensions*). But this loses "independence". On the contrary, we balance the variances of all *subspaces* (but each dimension is allowed to have a different variance). Driven by other motivations, the methods in [16], [8] allocate an adaptive number of bits to each principal component. On the contrary, our method allocates the principal components to each subspace.

Our derivation provides new theoretical explanations for the two criteria: they can be considered as minimizing the quantization distortion under a Gaussian distribution assumption.

3.2.6 A Parametric Model for Iterative Quantization

Interestingly, if we apply the Gaussian assumption to the ITQ objective (3), we can theoretically derive that the distortion of ITQ is minimized when the variances of the components are balanced. The balance can be done in the form of PCA followed by random rotation [9] or a Householder transform [3]. A recent method called Isotropic Hashing [28] explicitly optimizes the balance. The following derivations show a theoretical relation between balance and distortion.

Assume the data is subject to D-dimensional Gaussian $\mathcal{N}(0, \Sigma)$. The rotated data is subject to $\mathcal{N}(0, \hat{\Sigma})$ with $\hat{\Sigma} = R\Sigma R^{\mathrm{T}}$. Denote a diagonal element of $\hat{\Sigma}$ as $\hat{\sigma}_d^2$. The d-th dimension after rotation is subject to $\mathcal{N}(0, \hat{\sigma}_d^2)$. As discussed in Sec. 2.2.3, ITQ is using 2 codewords (with distance $2a$) to quantize each dimension. The distortion in this dimension is $E_d = \frac{1}{2}\int_0^{+\infty}(x-a)^2 p(x)dx + \frac{1}{2}\int_{-\infty}^{0}(x+a)^2 p(x)dx = a^2 - 2a\sqrt{\frac{2}{\pi}}\hat{\sigma}_d + \hat{\sigma}_d^2$, where $p(x)$ is the probability density function. Note this is the actual distortion rather than the lower bound. So the distortion of ITQ, under the Gaussian assumption, is given by:

$$E(R, a) = \sum_{d=1}^{D}\left(a^2 - 2a\sqrt{\frac{2}{\pi}}\hat{\sigma}_d + \hat{\sigma}_d^2\right). \quad (15)$$

Minimizing this distortion w.r.t. a gives us $a = \sqrt{\frac{2}{\pi}}\frac{1}{D}\sum_d \hat{\sigma}_d$. Putting a into (15) and omitting the constants, we obtain the following optimization problem:

$$\max_R \sum_d \hat{\sigma}_d, \quad s.t. \quad R^{\mathrm{T}}R = I. \quad (16)$$

This further leads us to:

$$\max_{\{\hat{\sigma}_d\}} \sum_d \hat{\sigma}_d, \quad s.t. \quad \sum_d \hat{\sigma}_d^2 = |\Sigma| = const. \quad (17)$$

This problem can be solved by a Lagrange multiplier: $\max_{\{\hat{\sigma}_d\}} \sum_d \hat{\sigma}_d + \lambda(|\Sigma| - \sum_d \hat{\sigma}_d^2)$. Computing the partial derivative of each $\hat{\sigma}_d$ we have $\hat{\sigma}_d = \frac{1}{2\lambda}$. This implies that $\hat{\sigma}_d$ should be equal to each other. In this case, the problem (16) becomes seeking an orthogonal matrix R such that the variances $\{\hat{\sigma}_d\}$ are balanced.

The above derivation indicates the balance criterion *alone* can give minimal distortion, if under these conditions: (i) the data is Gaussian; (ii) each subspace is one-dimensional and has two codewords; and (iii) the distance between the two codewords in any subspace is a constant (a).

3.3 Non-Parametric vs. Parametric – the Combinatorial Nature

The Eigenvalue Allocation algorithm also reveals the *combinatorial nature* of the problem. An orthogonal matrix R involves the issue of permutating the dimensions. A coordinate descent algorithm (like our non-parametric solution in Sec. 3.1) may not be good at solving combinatorial problems. Next we show by experiments that at least for a Gaussian distribution, the non-parametric solution can be sensitive to the initialization. And the parametric solution can be a better way for initialization.

We generate a synthetic dataset subject to a 128-d independent Gaussian distribution, where the variances are given by $\sigma_d^2 = e^{-0.1d}$ (d=1,...,128). This synthetic set has 1 million data points and 10000 queries. We fix $M = 4$ and $k = 256$. We test three ways of the non-parametric solution: (i) initialized by random rotation (denoted as $\mathrm{OPQ}_{\mathrm{NP+RR}}$), (ii) initialized by randomly ordering dimensions (denoted as $\mathrm{OPQ}_{\mathrm{NP+RO}}$), and (iii) initialized by Eigenvalue Allocation (simply denoted as $\mathrm{OPQ}_{\mathrm{NP}}$). We also show the result of the parametric solution ($\mathrm{OPQ}_{\mathrm{P}}$). The following table shows the distortion and the mAP (search for 100 NNs using ADC):

In this table the randomized algorithms are run 10 trials and averaged (shown with std). We can see that random initializations cannot approach the optimality achieved by $\mathrm{OPQ}_{\mathrm{P}}$ or $\mathrm{OPQ}_{\mathrm{NP}}$. We also see

	$\text{OPQ}_{\text{NP+RR}}$	$\text{OPQ}_{\text{NP+RO}}$	OPQ_{NP}	OPQ_{P}
E	2.324±0.004	2.371±0.063	**2.282**	**2.284**
mAP	0.169±0.001	0.156±0.015	**0.176**	**0.176**

TABLE 3

OPQ_{NP} can hardly improve OPQ_{P}. This indicates the non-parametric solution can be sensitive to the initializations. If no other prior knowledge is available, the parametric solution can be a reasonable way to initialize it.

In Sec. 4 we will see the impact of initializations to the real GIST dataset.

4 Experiments

We evaluate the performance of the optimized product quantization in three applications.

4.1 Compact Encoding for Approximate Distance Computation

In the first experiment, we study the performance of compact encoding for approximate distance computation as in [1]. This is a common exhaustive search strategy used by PQ [1] and other binary embedding methods like [9]. Given a budget of B bits per vector, each vector in the database is encoded as a B-bit code. This allows to fit millions/billions of data in the memory. In the on-line search, the data are ranked in the order of their approximate distances to the query. In [1] Symmetric Distance Computation (SDC) and Asymmetric Distance Computation (ADC) are proposed as two ways of approximating distances, depending on whether the query is encoded or not. We test both cases. Such exhaustive ranking is fast for million-scale data: *e.g.*, for $B = 64$ bits it takes 20 ms per 1 million distance computation (an Intel Core2 2.13GHz CPU using a single core, 8G RAM).

The SDC of ITQ [9] is equivalent to Hamming ranking as popularly used in other binary embedding methods [15], [16], [17], [18], [19], [20]. To apply ADC for ITQ, we need to find its codewords. This is straightforward in the formulation (3) which involves the length a. It can be shown (3) is quadratic on a (see also [9]) so a can be very simply solved by least squares. With a we can represent any codeword of ITQ. In this case ITQ is just like a special case of PQ with $M = B$ subspaces and $k = 2$ codewords. So the ADC of ITQ can also adopt lookup tables. We notice other ways of Asymmetric Hamming (AH) distance [29], [30] have been developed. We adopt the ADC of ITQ as described above because it is analogous to other quantization methods. We also find the ADC and AH of ITQ behave very similar in experiments.

We evaluate on three real datasets and one synthetic dataset. The first two datasets are SIFT1M and GIST1M [1]. SIFT1M consists of 1 million 128-d SIFT vectors [31] and 10k queries. GIST1M consists of 1 million 960-d GIST vectors [32] and 1k queries. The third dataset MNIST[5] consists of 70k images of handwritten digits, each as a 784-d vector concatenating all pixels. We randomly sample 1k as the queries and use the remaining as the data base. We further generate a synthetic dataset subject to a 128-d independent Gaussian distribution, where the variances are given by $\sigma_d^2 = e^{-0.1d}$ (d=1,...,128): this is a long-tail curve fitted to the eigenvalues of many real datasets. This synthetic set has 1 million data points and 10k queries.

We consider K Euclidean nearest neighbors as the true neighbors and have evaluated K=1 to 1000. We find the comparisons among the methods are nearly unchanged. In this work we report K=100.

4.1.1 Comparisons with Quantization Methods

We first compare with the following methods. All of them can be viewed as kinds of vector quantization:

- **OPQ_{P}**: this is our parametric solution.
- **OPQ_{NP}**: this is our non-parametric solution initialized by the parametric solution.
- **PQ_{RO}**: the dimensions are *randomly ordered* as suggested in [1].
- **PQ_{RR}**: the data are aligned using PCA and then *randomly rotated*, as suggested in [3]. We have also optimized the Householder transform as in [3] to balance the variances of the components. We find this is comparable with random rotation, as also reported in [3].
- **TC** (Transform Coding [8]): this is a Scalar Quantization (SQ) method. SQ is a special case of PQ that each dimension forms a subspace. TC uses the principal components as the subspaces. It assigns each principal component with an adaptive number of bits. A similar method was also concurrently proposed in [33].
- **ITQ** [9]: this is a special vector quantization method that is also binary embedding.

Notice that in these settings we have assumed there is no prior knowledge available. Later we will study the case with prior knowledge.

Given the code-length B, all the PQ-based methods (OPQ_{NP}, OPQ_{P}, PQ_{RO}, PQ_{RR}) assign 8 bits to each subspace ($k = 256$). The subspace number M is $B/8$.

Results in the synthetic dataset

Fig. 3 shows the performance on the synthetic Gaussian data. Here we evaluate by the recall vs. N, *i.e.*, the proportion of the true nearest neighbors ranked in the top N positions. We can see that OPQ_{NP} and OPQ_{P} perform almost the same. We verify that OPQ_{P} have achieved the theoretical minimum in (14) (6.314×10^{-3}). This implies that, under a Gaussian distribution, our parametric solution is optimal.

On the contrary, PQ_{RO} and PQ_{RR} perform substantially worse. In the Gaussian data, the PQ_{RR} performs

5. http://yann.lecun.com/exdb/mnist/

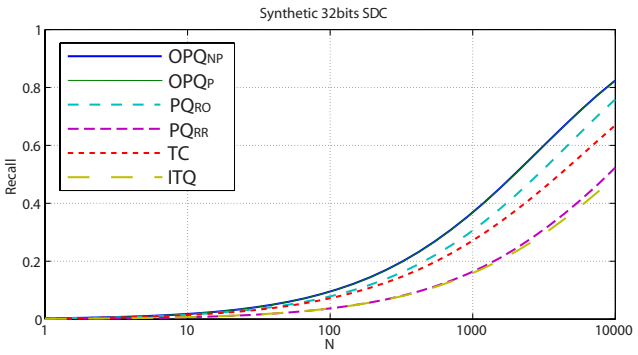


Fig. 3: Comparison on Gaussian synthetic data using Symmetric Distance Computation and 32-bit codes.

worse than ITQ. This indicates that the subspace decomposition can be very important to the performance of PQ, even under a simple Gaussian distribution. Besides, we find PQ_{RO} performs better than PQ_{RR}. This is because in the independent Gaussian distribution, PQ_{RO} automatically satisfies the "independence" criterion, and the random order can somewhat "balance" the variances of the subspaces.

Results in real datasets without prior knowledge

Next we evaluate the performance on real datasets and assume the prior knowledge is not available. We are particularly interested in the lack of prior knowledge, because we expect the methods to work well in general data that are unstructured, such as raw pixels or compressed representations (by PCA, sparse coding, *etc*.). Many previous works focus on the highly structured SIFT/GIST vectors and harness these structures. But this limits the investigation on general data.

All the above methods can be considered as somewhat blind to the prior knowledge. This is because the effects of the structures are weakened if the vectors undergo some PCA, random ordering, or random rotation.

In Fig. 4, 5, and 6 we compare the results on SIFT1M, GIST1M, and MNIST. We show the recall vs. N with B=64 bits using SDC/ADC (Fig. 4, 5, 6 (a)(b)), and the mAP vs. code length B using SDC (Fig. 4, 5, 6 (c)). We can also evaluate their quantization distortion vs. code length B (Fig. 4, 5, 6 (d)). More comparisons evaluated by different metrics are given in the supplementary materials.

We find our both solutions substantially outperform the existing methods. The superiority of our methods present on both SDC and ADC. In all cases even our simple parametric method OPQ_P has shown prominent improvement over PQ_{RO} and PQ_{RR}. This again indicates that PQ-based methods strongly depend on the space decomposition. We also notice the performance of PQ_{RR} is disappointing. Although

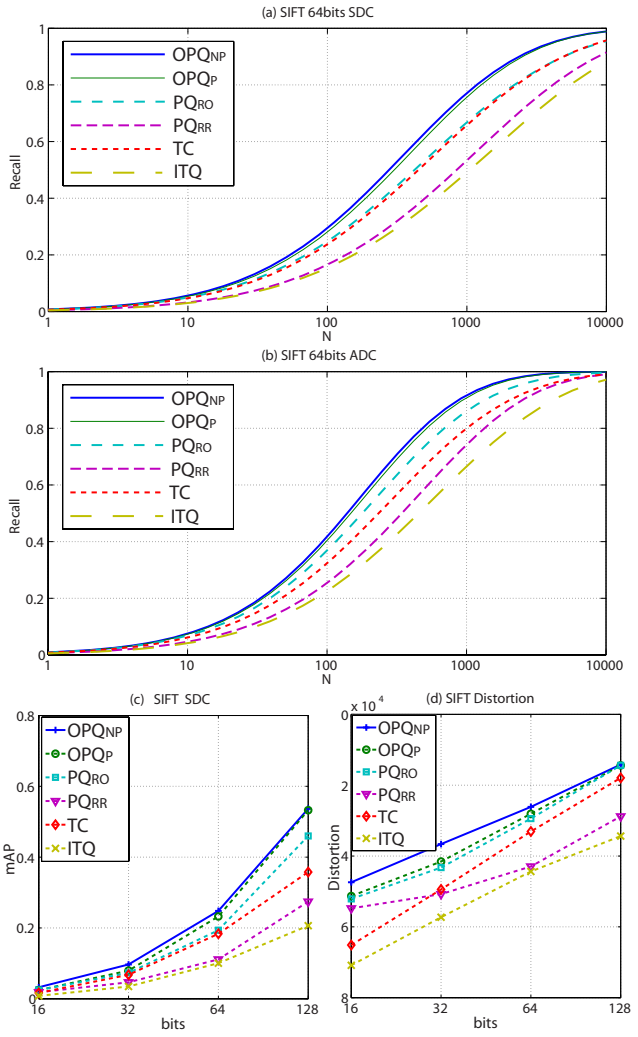


Fig. 4: Comparisons on SIFT1M. (a)(b): recall at the N top ranked samples, using SDC/ADC and 64-bit codes. (c): mean Average Precision vs. code-length using SDC. (d): distortion vs. code-length.

this method (and the Householder transform in [3]) can balance the variance using a random rotation, the independence between subspaces is lost in the random rotation.

Our non-parametric solution OPQ_{NP} further improves the results of the parametric solution OPQ_P in the SIFT1M and MNIST datasets. This is because these two datasets exhibit non-Gaussian distributions: the SIFT1M set has two distinct clusters (this can be visualized by plotting the first two principal components of SIFT), and MNIST can be expected to have 10 clusters. In these very non-Gaussian cases, the parametric OPQ_P is certainly not optimal, and the non-parametric OPQ_{NP} is able to further reduce the distortion. In GIST1M our two methods OPQ_{NP} and OPQ_P are comparable.

We notice that TC performs clearly better than PQ_{RO} and PQ_{RR} in the GIST1M set. But TC is inferior

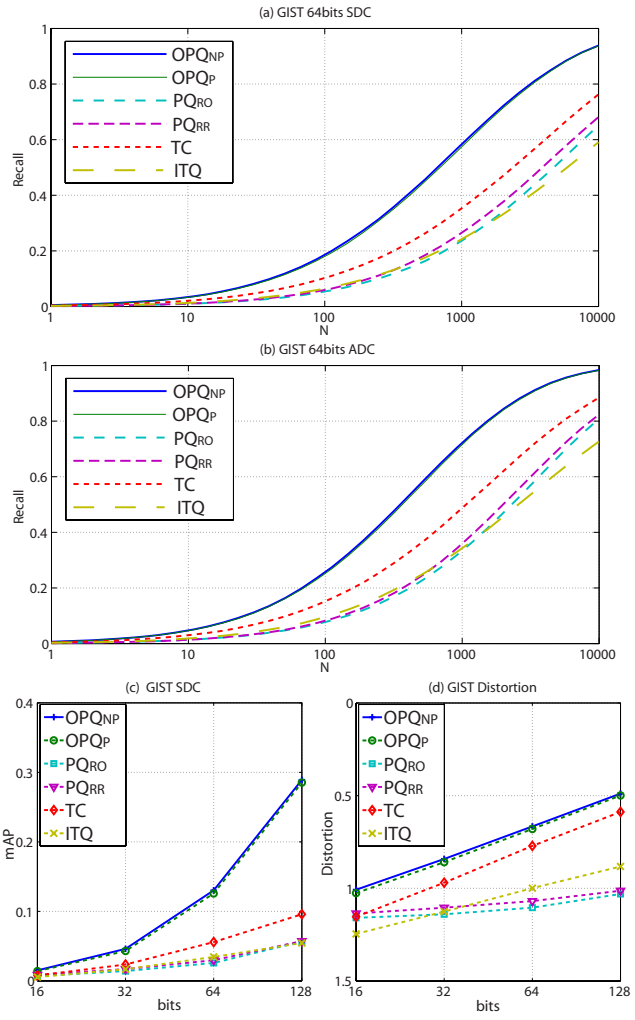


Fig. 5: Comparisons on GIST1M.

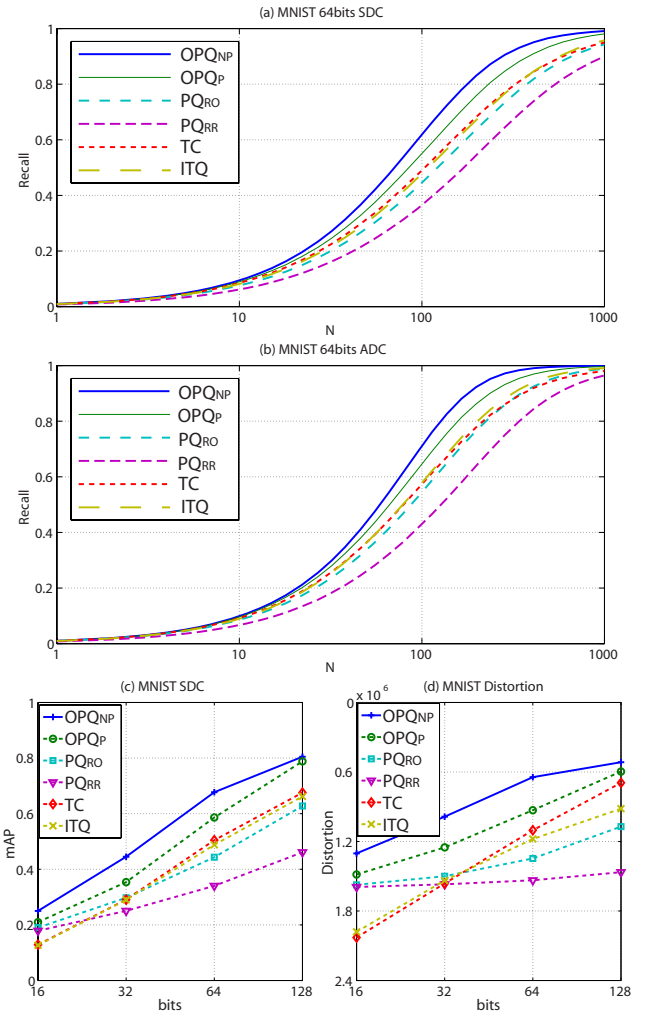


Fig. 6: Comparisons on MNIST.

to our methods in all datasets. This is because TC is scalar quantization, while our method quantizes multi-dimensional subspaces. Further, TC assigns an adaptive number of bits to each eigenvalue, while our method assigns the eigenvalues to each subspace. Since bit numbers are discrete but eigenvalues are continuous, it is easier for our method to achieve balance.

Results in real datasets with prior knowledge

In [1] it has been noticed that PQ works much better if utilizing the prior knowledge that SIFT and GIST are concatenated histograms. The so-called "natural" order is that each subspace consists of neighboring histograms. The "structural" order (when $M = 8$) is that each subspace consists of the same bin of all histograms (each histogram has 8 bins). We denote PQ with priori knowledge as PQ_{nat} (natural) and PQ_{str} (structural). Note such priors may limit the choices of M and B.

In Fig. 7 we compare PQ_{pri} with our prior-free non-parametric method OPQ_{NP}. We also evaluate our non-parametric method using the prior orders as initialization, denoted as $\text{OPQ}_{\text{NP+nat}}$ and $\text{OPQ}_{\text{NP+str}}$. In Fig. 7 we only show the better order (SIFT: natural; GIST: structural). We see even our prior-free method OPQ_{NP} outperforms the prior-based PQ on both sets. In SIFT1M our prior-dependent $\text{OPQ}_{\text{NP+nat}}$ improves further thanks to a better initialization. In GIST1M our $\text{OPQ}_{\text{NP+str}}$ is slightly inferior to our prior-free OPQ_{NP}.

Sensitivity to Initializations

To further see the impact of initialization, we evaluate the OPQ_{NP} (initialized by Eigenvalue Allocation), $\text{OPQ}_{\text{NP+str}}$, and $\text{OPQ}_{\text{NP+nat}}$ on GIST1M. We also evaluate OPQ_{NP} initialized by a random rotation, denoted as $\text{OPQ}_{\text{NP+RR}}$. Table 4 shows the mAP on GIST1M using 64bits and ADC:

	$\text{OPQ}_{\text{NP+str}}$	$\text{OPQ}_{\text{NP+nat}}$	$\text{OPQ}_{\text{NP+RR}}$	OPQ_{NP}
mAP	0.191	0.182	0.190±0.003	**0.205**

TABLE 4: mAP on GIST1M (64bits, ADC)

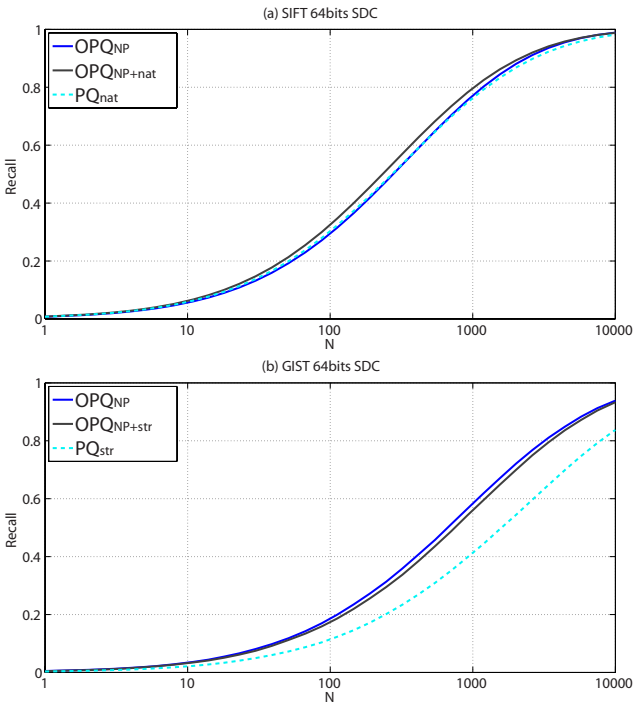


Fig. 7: Comparisons using prior knowledge. (a): SIFT1M. (b): GIST1M. Here the results are with 64 bits and SDC.

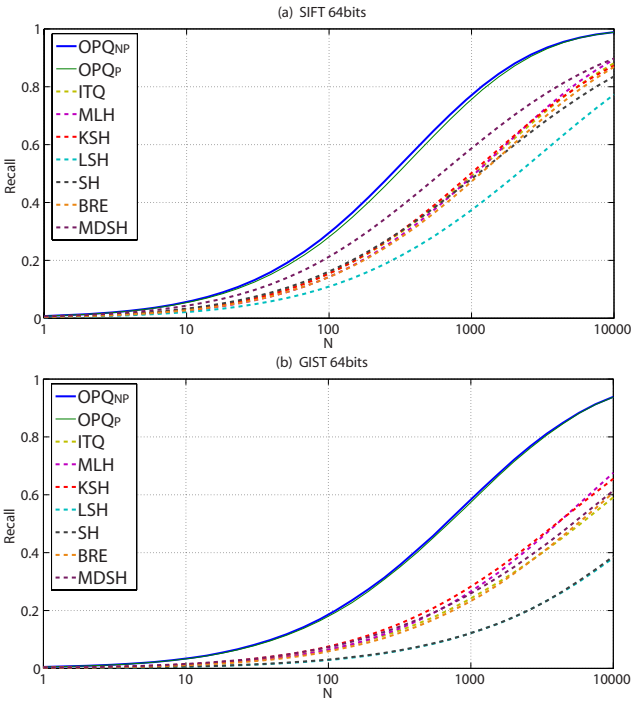


Fig. 8: Comparisons with binary embedding methods using 64 bits. (a): SIFT1M. (b): GIST1M.

We see that non-parametric solution initialized by structural/natural orders or random rotation performs very similarly. This is also observed in the "Cartesian k-means" paper [13] ("Cartesian k-means" and our non-parametric solution are equivalent to each other if using the same initialization). However, we find our OPQ_{NP} (initialized by Eigenvalue Allocation) performs better than the other three. This indicates our non-parametric solution (also Cartesian k-means) relies on the initializations. Our Eigenvalue Allocation provides a better initialization in the GIST dataset.

4.1.2 *Comparisons with Binary Embedding Methods*

Binary embedding is a popular way of encoding vectors [15], [16], [17], [27], [9], [18], [19]. For nearest neighbor search, one can rank the encoded data vectors by their Hamming distance to the encoded query. Not all binary embedding methods (except ITQ or orthogonal ones) can be formulated as vector quantization (encoding/decoding) in Sec. 2.1, because these binary methods only have partition boundaries but no codeword.

We compare with the following binary embedding methods: Locality Sensitive Hashing (LSH) [15], Spectral Hashing (SH) [16], Binary Reconstructive Embedding (BRE) [17], Minimal Loss Hashing (MLH) [18], and Kernel-based Supervised Hashing (KSH) [19]. We also compare with Multidimensional Spectral Hashing (MDSH) [34], which uses weighted Hamming distances.

Fig. 8 shows the comparisons on SIFT1M/GIST1M using 64 bits. For fair comparisons, our methods use SDC here. We see our OPQ_{NP} and OPQ_{P} substantially outperform these binary embedding methods.

4.2 Building Inverted Multi-Index for Non-exhaustive Search

The inverted multi-index method [2] uses a product quantizer for inverted indexing. Our optimized product quantizer can improve the performance of the resulted inverted indexing.

We briefly introduce the method of [2] as follows. To generate a fine codebook with k^M codewords, this method applies a product quantizer using M subspaces with k sub-codeword in each. Unlike [1] that uses this codebook to encode the data, this method uses it to build inverted indexing. Off-line, each codeword has been assigned a short list that contains all the data vectors belonging to this codeword (*i.e.*, nearest to it). On-line, a query will find a number of nearest codewords and retrieve all their short lists. Although there are essentially k^M codewords in the original space, the distance of the query to these codeword can be given by M 1-by-k lookup tables (analogous to ADC). The recommended number M is 2 in [2]. In this case, the nearest codewords to the query can be obtained by a priority queue in a $k \times k$ table spanned by two 1-by-k tables.

This inverted multi-index method is a current state-of-the-art non-exhaustive search method. Because this

methods	T	R@1	R@10	R@100	time(ms)
Multi-D-ADC	10000	$0.327_{(0.304)}$	$0.681_{(0.665)}$	$0.748_{(0.740)}$	$6.8_{(7)}$
OMulti-D-OADC	10000	**0.345**	**0.725**	**0.794**	6.9
Multi-D-ADC	30000	$0.332_{(0.328)}$	$0.774_{(0.757)}$	$0.885_{(0.885)}$	$16.9_{(16)}$
OMulti-D-OADC	30000	**0.366**	**0.807**	**0.913**	16.9
Multi-D-ADC	100000	$0.344_{(0.334)}$	$0.809_{(0.793)}$	$0.960_{(0.959)}$	$52.0_{(49)}$
OMulti-D-OADC	100000	**0.373**	**0.841**	**0.973**	51.5

TABLE 5: Comparisons with inverted multi-index [2] in **SIFT1B**, with ADC-based re-ranking. This table corresponds to Table 1 in [2]. The numbers in the brackets are reported in [2]. The time is the average per query. It consists of the short lists retrieval time and the ranking time.

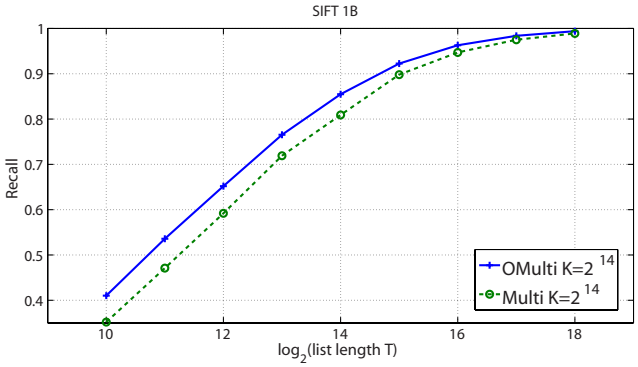


Fig. 9: OPQ for inverted multi-index [2]. Here the original inverted multi-index [2] is termed as "Multi", and our optimized PQ for inverted multi-index is termed as "OMulti". This figure corresponds to Figure 3 (left) in [2].

4096-d float	$0.564_{(0.556)}$	
PCA $\rightarrow$ 128-d float	$0.551_{(0.557)}$	
	PQ_{RR}	OPQ_{NP}
4 bytes	0.260	**0.377**
8 bytes	0.381	**0.477**
16 bytes	0.479	**0.522**
32 bytes	0.530	**0.543**

TABLE 6: mAP in Holiday using VLAD. Here we use the improved version of VLAD in [35]. The numbers in the brackets are reported in [35].

4096-d float	$0.588_{(0.595)}$	
PCA $\rightarrow$ 128-d float	$0.561_{(0.565)}$	
	PQ_{RR}	OPQ_{NP}
4 bytes	0.272	**0.384**
8 bytes	0.370	**0.455**
16 bytes	0.486	**0.519**
32 bytes	0.530	**0.551**

TABLE 7: mAP in Holiday using Fisher Vectors. The numbers in the brackets are reported in [35].

method involves a product quantizer, it is straightforward to apply for our method to optimize this product quantizer. Following the experiment settings in [2], we study the ANN performance on the SIFT1B [1] dataset containing 1 billion SIFT. Here we consider the first nearest neighbor as the ground truth. We optimize a product quantizer using our non-parametric solution initialized by the natural order. Fig. 9 shows the recall vs. T, where T is the total number of retrieved vectors in the short lists. We test $M = 2$ and $k = 2^{14}$ as in [2]. We see that our optimized quantizer improves the recall of the retrieval.

In [2], the retrieved vectors can be re-ranked using PQ as a compact encoding method. This is termed as "Multi-D-ADC" in [2]. Notice the product quantizer used to build the inverted indexing and the one used for compact encoding are different. It is straightforward to apply for our method to optimize both quantizers. Our result is termed as "**O**Multi-D-**O**ADC" to highlight the two optimized quantizers.

Table 5 shows the recall vs. the N top re-ranked vectors, together with the querying time. We see our optimized quantizers can improve the accuracy. To the best of our knowledge, in terms of both accuracy and speed, this is the state-of-the-art ANN solution to this billion-scale dataset.

4.3 Compacting Image Representations for Image Retrieval

In [3], PQ was introduced as a way of compacting image representations for image retrieval. In this scenario, the local descriptors of an image are first aggregated as a high-dimensional (often thousands of dimensions) vector. The aggregation methods include the Fisher Kernel [11] and VLAD [3], [35]. The aggregated vector is normalized and compressed by PCA. The compressed vector is then compacted into a short code by PQ for retrieval.

Unlike SIFT/GIST, the Fisher vectors and the VLAD vectors do not present structural orders, especially due to the PCA. In [3], [35] it is proposed to apply a random rotation matrix before PQ (*i.e.*, PQ_{RR}). Instead,

we apply our optimized product quantizer OPQ_{NP} for this compact encoding task.

Table. 6 and Table. 7 show the retrieval accuracy in the Holiday dataset [3]. We only simply replace the PQ_{RR} by OPQ_{NP} for compact encoding. For both VLAD and Fisher Vectors we study 4096-d uncompressed features. These features are first reduced to 128-d by PCA. Then we apply PQ_{RR}/OPQ_{NP} on these 128-d features. We can see our optimized product quantizer significantly improves the retrieval accuracy of PQ_{RR}.

5 Discussion and Conclusion

We have proposed two solutions to optimized product quantization. Because PQ has witnessed many applications in computer vision, and also because the space decomposition has great impacts on the PQ performance, we believe this work has made PQ a more practical and powerful method for many applications.

References

[1] H. Jegou, M. Douze, and C. Schmid, "Product quantization for nearest neighbor search," *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, vol. 33, 2011.

[2] A. Babenko and V. S. Lempitsky, "The inverted multi-index," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012, pp. 3069–3076.

[3] H. Jegou, M. Douze, C. Schmid, and P. Perez, "Aggregating local descriptors into a compact image representation," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2010, pp. 3304–3311.

[4] J. Sivic and A. Zisserman, "Video google: a text retrieval approach to object matching in videos," in *IEEE International Conference on Computer Vision (ICCV)*, 2003.

[5] O. Boiman, E. Shechtman, and M. Irani, "In defense of nearest-neighbor based image classification," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2008.

[6] A. B. Torralba, R. Fergus, and W. T. Freeman, "80 million tiny images: A large data set for nonparametric object and scene recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, vol. 30, pp. 1958–1970, 2008.

[7] R. M. Gray and D. L. Neuhoff, "Quantization," *IEEE Transactions on Information Theory (TIT)*, 1998.

[8] J. Brandt, "Transform coding for fast approximate nearest neighbor search in high dimensions," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2010.

[9] Y. Gong and S. Lazebnik, "Iterative quantization: A procrustean approach to learning binary codes," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2011.

[10] D. Nister and H. Stewenius, "Scalable recognition with a vocabulary tree," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2006.

[11] F. Perronnin and C. Dance, "Fisher kernels on visual vocabularies for image categorization," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2007.

[12] T. Ge, K. He, Q. Ke, and J. Sun, "Optimized product quantization for approximate nearest neighbor search," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013.

[13] M. Norouzi and D. Fleet, "Cartesian k-means," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013.

[14] J. B. MacQueen, "Some methods for classification and analysis of multivariate observations," in *Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability*. University of California Press, 1967, pp. 281–297.

[15] P. Indyk and R. Motwani, "Approximate nearest neighbors: towards removing the curse of dimensionality," in *ACM Symposium on Theory of Computing (STOC)*, 1998, pp. 604–613.

[16] Y. Weiss, A. Torralba, and R. Fergus, "Spectral hashing," in *Advances in Neural Information Processing Systems (NIPS)*, 2008, pp. 1753–1760.

[17] B. Kulis and T. Darrell, "Learning to hash with binary reconstructive embeddings," vol. 22, 2009, pp. 1042–1050.

[18] M. E. Norouzi and D. J. Fleet, "Minimal loss hashing for compact binary codes," in *International Conference on Machine Learning (ICML)*, 2011, pp. 353–360.

[19] W. Liu, J. Wang, R. Ji, Y.-G. Jiang, and S.-F. Chang, "Supervised hashing with kernels," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[20] K. He, F. Wen, and J. Sun, "K-means Hashing: an Affinity-Preserving Quantization Method for Learning Binary Compact Codes," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013.

[21] P. Schönemann, "A generalized solution of the orthogonal procrustes problem," *Psychometrika*, vol. 31, no. 1, pp. 1–10, 1966.

[22] T. Cover and J. Thomas, *Elements of information theory*. John Wiley & Sons, Inc., 1991, ch. 13, p. 348.

[23] Z. Wen and W. Yin, "A feasible method for optimization with orthogonality constraints," *Mathematical Programming*, pp. 1–38, 2010.

[24] A. Cauchy, *Cours d'analyse de l'École Royale Polytechnique*. Imprimerie royale, 1821.

[25] R. A. Horn and C. R. Johnson, *Matrix Analysis*. Cambridge University Press, 1990, ch. 7, p. 478.

[26] S. Mertens, "The easiest hard problem: Number partitioning," *Computational Complexity and Statistical Physics*, vol. 125, no. 2, 2006.

[27] J. Wang, S. Kumar, and S.-F. Chang, "Semi-supervised hashing for scalable image retrieval," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2010.

[28] W. Kong and W.-J. Li, "Isotropic hashing," in *Advances in Neural Information Processing Systems (NIPS)*, 2012, pp. 1655–1663.

[29] W. Dong, M. Charikar, and K. Li, "Asymmetric distance estimation with sketches for similarity search in high-dimensional spaces," in *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, 2008.

[30] A. Gordo, F. Perronnin, Y. Gong, and S. Lazebnik, "Asymmetric distances for binary embeddings," 2013.

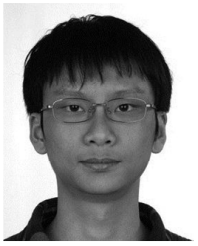
[31] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision (IJCV)*, vol. 60, pp. 91–110, 2004.

[32] A. Oliva and A. Torralba, "Modeling the shape of the scene: a holistic representation of the spatial envelope," *International Journal of Computer Vision (IJCV)*, 2001.

[33] H. Sandhawalia and H. Jégou, "Searching with expectations," in *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*. IEEE, 2010, pp. 1242–1245.

[34] Y. Weiss, R. Fergus, and A. Torralba, "Multidimensional spectral hashing," in *European Conference on Computer Vision (ECCV)*, 2012.

[35] H. Jegou, F. Perronnin, M. Douze, J. Sanchez, P. Perez, and C. Schmid, "Aggregating local image descriptors into compact codes," *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, vol. 34, no. 9, pp. 1704–1716, 2012.



Tiezheng Ge is a PhD candidate in the Department of Electronic Science and Technology, University of Science and Technology of China. Before that, he received the BS degree from University of Science and Technology of China in 2009. He is currently working as an intern of Microsoft Research Asia. His research interest is large scale image search.

Kaiming He is currently a researcher at Microsoft Research Asia. He received the BS degree from Tsinghua University in 2007, and the PhD degree from the Chinese University of Hong Kong in 2011. He joined Microsoft Research Asia in 2011. His research interests include computer vision and computer graphics. He has won the Best Paper Award at the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) 2009.

Qifa Ke is currently a software development engineer in Bing, working on search relevance. Before joining Bing, he was a Researcher in Microsoft Research Silicon Valley lab, working on Internet image and video search, large-scale data analysis and machine learning, and data-parallel distributed computing systems. Qifa Ke received his Ph.D. in Computer Science from Carnegie Mellon University in 2003.”.

Jian Sun is currently a principal researcher at Microsoft Research Asia. He got the BS degree, MS degree and Ph.D. degree from Xian Jiaotong University in 1997, 2000 and 2003. He joined Microsoft Research Asia in July, 2003. His current two major research interests are interactive computer vision (user interface + vision) and internet computer vision (large image collection + vision). He is also interested in stereo matching and computational photography. He has won the Best Paper Award at the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) 2009.