Frontiers of Information Technology & Electronic Engineering
www.jzus.zju.edu.cn; engineering.cae.cn; www.springerlink.com
ISSN 2095-9184 (print); ISSN 2095-9230 (online)
E-mail: jzus@zju.edu.cn

Review:

Vector quantization: a review*

Ze-bin WU[1], Jun-qing YU‡[1,2]

[1]*Department of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan 430074, China*

[2]*Center of Network and Computation, Huazhong University of Science and Technology, Wuhan 430074, China*

E-mail: zbwu@hust.edu.cn; yjqing@hust.edu.cn

Received Dec. 12, 2017; Revision accepted Mar. 17, 2018; Crosschecked Apr. 11, 2019

Abstract: Vector quantization (VQ) is a very effective way to save bandwidth and storage for speech coding and image coding. Traditional vector quantization methods can be divided into mainly seven types, tree-structured VQ, direct sum VQ, Cartesian product VQ, lattice VQ, classified VQ, feedback VQ, and fuzzy VQ, according to their codebook generation procedures. Over the past decade, quantization-based approximate nearest neighbor (ANN) search has been developing very fast and many methods have emerged for searching images with binary codes in the memory for large-scale datasets. Their most impressive characteristics are the use of multiple codebooks. This leads to the appearance of two kinds of codebook: the linear combination codebook and the joint codebook. This may be a trend for the future. However, these methods are just finding a balance among speed, accuracy, and memory consumption for ANN search, and sometimes one of these three suffers. So, finding a vector quantization method that can strike a balance between speed and accuracy and consume moderately sized memory, is still a problem requiring study.

Key words: Approximate nearest neighbor search; Image coding; Vector quantization
https://doi.org/10.1631/FITEE.1700833 **CLC number:** TP391

1 Introduction

Vector quantization (VQ) was initially called block codes or block quantization, and was proposed for speech coding to reduce the consumption of bandwidth. Vector quantization is a branch of quantization, which aims to compress image or speech data, and VQ is a generalization of scalar quantization.

Scalar quantization divides a quantity into a discrete number of small parts, each of which is assumed to be an integral multiple of a common quantity (Gray and Neuhoff, 1998). A scalar quantizer $q(x)$ is a function defined by a set of intervals or cells $S=\{S_i|i\in I\}$ (I is the index set that consists of consecutive integers), and a set of reproduction values, one reproduction value for each cell. Thus,

$$q(x)=c_i,\ x\in S_i. \tag{1}$$

For a quantizer, the reproduction set C is much smaller than the size of the input space of x, which means that $q(x)$ maps a large set into a much smaller set C, the values of which will be used to represent x, saving a lot of bandwidth needed to transmit and a lot of memory needed to store the raw data, especially image data. The quality of a quantizer can be described by a distortion measure $d(x,\hat{x})$, which results from reproducing x by $\hat{x}$, and the goal of scalar quantization is to encode the input data x with as few bits as possible according to Shannon's rate-distortion theory or source coding with a fidelity criterion (Shannon, 1948, 1959).

Scalar quantization can be divided into fixed-rate quantization and variable-rate quantization,

‡ Corresponding author

* Project supported by the National Natural Science Foundation of China (Nos. 61572211, 61173114, and 61202300)

ORCID: Jun-qing YU, http://orcid.org/0000-0001-7057-0402

according to whether the lengths of bits used to encode the codewords (namely the reproductions) are the same. For fixed-rate quantization, the rate of code in bit per codeword is $\log_2 k$, where k is the number of codewords. Though it is very simple and convenient to assign equal bits to each codeword, it is wasteful if cells S_i have different probabilities. According to Shannon's source coding theory, the size of codes needed to encode the input data can be reduced if shorter codes are assigned to higher probability cells, leading to variable-rate quantization. Usually, there exists redundancy resulting from the correlation or dependence among input data, which should be removed to compress the data. Two popular methods of this kind are predictive coding and transform coding. However, despite the effectiveness and practicability of scalar quantization techniques, they cannot make sufficient use of the redundancy in the data. Thus, VQ is put forward.

As pointed out by Shannon, VQ can achieve the optimal rate distortion performance with a fidelity criterion in contrast to scalar quantization. VQ maps the large input vector space to a much smaller vector space corresponding to a codebook and is a good way to compress the vectors into compact binary codes. Thus, VQ is a very effective way to save bandwidth and storage for transmitting and storing speech and image if used for speech coding and image coding. Many VQ methods have been developed, like tree-searched VQ (Buzo et al., 1980), multiple stage VQ (Juang and Gray, 1982), product VQ (Sabin and Gray, 1982, 1984; Gray and Neuhoff, 1998; Jégou et al., 2010), lattice VQ (Gersho, 1979), mean-removed VQ (Gray, 1984; Gersho and Gray, 1991; Gray and Neuhoff, 1998), and feedback VQ (Kieffer, 1982).

Over the past decade, quantization-based approximate nearest neighbor (ANN) search has been developing very fast and many methods have emerged for searching images with codes in the memory for large-scale datasets. Among them are product quantization (Jégou et al., 2010), residual VQ (Ai et al., 2014), joint inverted-index (Xia et al., 2013), and additive quantization (Babenko and Lempitsky, 2014). Their most impressive characteristics are the use of multiple codebooks and this may well continue into the future. However, these methods are just finding a balance among speed, accuracy, and memory consumption for ANN search; sometimes one of these three suffers. So, finding a VQ method that can strike a balance among speed, accuracy, and memory is still a problem requiring study.

2 Vector quantization (VQ)

Vector quantization (or block quantization or multidimensional quantization) was first described in Shannon (1959) as block source coding, which was described to be an approach that can achieve optimal rate distortion performance. VQ (Gray, 1984; Gersho and Gray, 1991; Gray and Neuhoff, 1998) is characterized by a d-dimensional partition, which is described by disjoint cells S_i and a d-dimensional codebook C, namely the set of reproduction points or codevectors or codewords, and quantizes all dimensions simultaneously. Vector quantizer Q can be defined as

$$Q : \mathbb{R}^d \to C, \tag{2}$$

where $C = \{\boldsymbol{c}_i | i \in I,\ \boldsymbol{c}_i \in \mathbb{R}^d\}$ is composed of k vectors. The code rate of a quantizer is $r = (\log_2 k)/d$, which is the number of bits per vector component used to encode the input vector. A vector quantizer can be decomposed into two parts, the vector encoder and the vector decoder (Gersho and Gray, 1991). Encoder E maps from $\mathbb{R}^d$ to index set I and decoder D maps index set I into reproduction set C, namely the codebook. Thus,

$$E : \mathbb{R}^d \to I, \quad D : I \to C. \tag{3}$$

As can be seen from Eq. (3), the encoder determines to which partition the input vector will belong and the decoder outputs the reproduction value or codeword corresponding to the index of the partition. The overall operation of VQ is defined as

$$Q(\boldsymbol{x}) = D(E(\boldsymbol{x})). \tag{4}$$

Now consider that $\boldsymbol{C}$ is a matrix, where the i^{th} codeword $\boldsymbol{c}_i$ is the i^{th} column of $\boldsymbol{C}$, so that $\boldsymbol{C} = [\boldsymbol{c}_1, \boldsymbol{c}_2, \ldots, \boldsymbol{c}_k] \in \mathbb{R}^{d\times k}$, where $\boldsymbol{c}_i$ is a d-dimensional column vector. Then we have

$$Q(\boldsymbol{x}) = \boldsymbol{C}\boldsymbol{b}, \tag{5}$$

where $\boldsymbol{b} = [b_1, b_2, \ldots, b_k]^{\mathrm{T}}$ and b_i=1 if $\boldsymbol{x} \in S_i$; otherwise, b_i=0. As can be seen, vector $\boldsymbol{b}$ is a selector that determines which codeword will be used to represent $\boldsymbol{x}$. The quality of a quantizer can be measured by

the mean squared error (MSE) between input vector $\boldsymbol{x}$ and corresponding reproduction $\boldsymbol{c}_i$:

$$\sum_{i=1}^{N} \|\boldsymbol{x}_i - \boldsymbol{C}\boldsymbol{b}_i\|_2^2, \tag{6}$$

where $\|\cdot\|_2$ is the ℓ_2 norm, N the number of input vectors, $\boldsymbol{x}_i$ the i^{th} input vector, and $\boldsymbol{b}_i$ the corresponding selector. Let $\boldsymbol{X} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N]$ and $\boldsymbol{B} = [\boldsymbol{b}_1, \boldsymbol{b}_2, \ldots, \boldsymbol{b}_N]$. Then Eq. (6) can be written as

$$\text{MSE} = \|\boldsymbol{X} - \boldsymbol{C}\boldsymbol{B}\|_{\text{F}}^2, \tag{7}$$

where $\|\cdot\|_{\text{F}}$ denotes the Frobenius norm. The object of the quantizer is to find the best $\boldsymbol{B}$ to minimize MSE. The objective function can be defined as

$$\begin{aligned} &\boldsymbol{B} = \arg\min_{\boldsymbol{B}} \|\boldsymbol{X} - \boldsymbol{C}\boldsymbol{B}\|_{\text{F}}^2 \\ &\text{s.t.}\ \ \boldsymbol{X} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N] \in \mathbb{R}^{d\times N}, \\ &\qquad \boldsymbol{C} = [\boldsymbol{c}_1, \boldsymbol{c}_2, \ldots, \boldsymbol{c}_k] \in \mathbb{R}^{d\times k}, \\ &\qquad \boldsymbol{B} = [\boldsymbol{b}_1, \boldsymbol{b}_2, \ldots, \boldsymbol{b}_N] \in \mathbb{R}^{k\times N}, \\ &\qquad \boldsymbol{b}_i \in \{0,1\}^k,\ \|\boldsymbol{b}_i\|_1 = 1,\ i = 1, 2, \ldots, N, \end{aligned} \tag{8}$$

where $\|\cdot\|_1$ denotes the ℓ_1 norm.

3 Optimal vector quantizer

An optimal vector quantizer is a quantizer that minimizes the mean squared error. According to Gersho (1982) and Lloyd (1982), an optimal vector quantizer should satisfy two conditions called the Lloyd optimality conditions. First, input vector $\boldsymbol{x}$ must be quantized to its nearest codeword $\boldsymbol{c}_i$. In terms of Euclidean distance, the condition can be defined as

$$Q(\boldsymbol{x}) = \arg\min_{\boldsymbol{c}_i \in \boldsymbol{C}} \|\boldsymbol{x} - \boldsymbol{c}_i\|_2^2. \tag{9}$$

As a result, a nearest neighbor partition is achieved. This is called the Voronoi partitions and the regions of partitions are called the Voronoi cells or Dirichlet cells. Each cell has a corresponding codeword. Second, the reproduction value, namely the codeword, must be the conditional mean of its corresponding Voronoi cell and this condition can be described as

$$\boldsymbol{c}_i = \mathbb{E}_{\boldsymbol{x}}[\boldsymbol{x}|\boldsymbol{x} \in R_i], \tag{10}$$

where R_i is the Voronoi cell corresponding to codeword $\boldsymbol{c}_i$, which is called the centroid. The optimal quantizer, which satisfies the Lloyd optimality conditions, is called the Lloyd quantizer or Voronoi quantizer, corresponding to the k-means clustering algorithm. As a result, the k-means algorithm is commonly used to generate the codebook $\boldsymbol{C}$.

4 Methods of vector quantization

According to whether the output of the quantizer is correlated to the previous input, traditional VQ methods can be divided into memoryless VQ methods and methods with memory. According to the codebook structure, traditional VQ methods can be divided into six types: tree-searched VQ (TSVQ) with a tree-structured codebook, multiple stage VQ (MSVQ) with a direct sum codebook, product VQ (PQ) with a Cartesian product codebook, lattice VQ (LVQ) with a lattice codebook, classified VQ with many sub-codebooks corresponding to the categories, and feedback VQ with a feedback codebook which is time-varying. The code book structure of fuzzy VQ (FVQ) is very common and what is different is the codebook construction procedure, so FVQ is not listed in Table 1.

Traditional VQ methods mostly originate from speech coding to reduce bandwidth consumption, focus on low-dimensional vectors, and operate directly on the pixels instead of descriptors like scale invariant feature transform (SIFT) (Lowe, 2004) extracted from images. However, with the development of VQ, improved versions have been proposed. These are effective for quantization-based ANN search (Sivic and Zisserman, 2003) on high-dimensional SIFT vectors, alleviating the curse of dimensionality problem (Beyer et al., 1999).

Table 1 is a summary of VQ, including the newly proposed VQ methods for ANN search, which are used mainly for image descriptors like SIFT, bag of feature (BOF) (Sivic and Zisserman, 2003), fisher vector (FV) (Perronnin and Dance, 2007), vector of locally aggregated descriptors (VLAD) (Jégou et al., 2012), and GIST (Oliva and Torralba, 2001), and have been seen as the state-of-the-art for ANN search. As can be seen from Table 1, VQ methods now can be divided into eight types instead of five types because of the appearance of two new kinds of codebook structure: linear combination codebook and joint codebook. The most extensively studied four VQ methods are tree-structured VQ, direct

Table 1 Classification of vector quantization (VQ) methods

Codebook structure	VQ method
Tree	TSVQ (Buzo et al., 1980) HKM (Nister and Stewenius, 2006) RPT (Dasgupta and Freund, 2009) TQ (Babenko and Lempitsky, 2015)
Lattice	LVQ (Gersho, 1979) PVQ (Fischer, 1986)
Classified	CVQ (Ramamurthi and Gersho, 1986) QCVQ (Chen et al., 2014)
Feedback	Feedback VQ (Kieffer, 1982) FSVQ (Foster et al., 1985)
Direct sum	MSVQ/RVQ (Juang and Gray, 1982) ERVQ (Ai et al., 2014) PRVQ (Wei et al., 2014) RVQ-NP (Guo et al., 2016) GRVQ (Liu et al., 2017)
Cartesian product	PQ (Jégou et al., 2010) TC (Brandt, 2010) OPQ (Ge et al., 2013) CKM (Norouzi and Fleet, 2013) DPQ (Heo et al., 2014) LOPQ (Kalantidi and Avrithis, 2014) OCKM (Wang et al., 2014) PTQ (Yuan and Liu, 2015a) KSQ (Ozan et al., 2016a)
Joint	JII (Xia et al., 2013)
Linear combination	AQ (Babenko and Lempitsky, 2014) CQ (Zhang T et al., 2014) SCQ (Zhang et al., 2015) TQ (Babenko and Lempitsky, 2015) LSQ (Martinez et al., 2016) CompQ (Ozan et al., 2016b)

sum VQ, Cartesian product VQ, and linear combination VQ. These VQ methods have been proposed for their good performance on ANN search application and can efficiently strike a balance among accuracy, speed, and memory consumption. The time and storage costs of these four most commonly used ANN search VQ methods are presented in Table 2. As can be seen from Table 2, tree-structured VQ is very fast. However, its storage cost is large, including the cost for all the nodes in the tree. In fact, tree-structured VQ is not so accurate for ANN search as the other three methods for using a space-partition tree. Product VQ is another fast method, achieved by splitting the vectors, and the storage cost is just the sum of the sub-codebooks. Direct sum VQ and linear combination VQ are relatively slow and the storage cost is nearly m (the number of sub-codebooks) times higher than that of PQ for the same codebook size because linear combination VQ performs quantization on the raw vectors without splitting.

Table 2 Time and storage costs of the most commonly used vector quantization (VQ) methods for approximate nearest neighbor (ANN) search

Codebook structure	VQ method	Encoding time	Storage cost
Tree	HKM	$O(D\log_2 K)$	$O(PDK)$
Direct sum	RVQ	$O(D(\sum_{i=1}^{L} k_i))$	$O(D(\sum_{i=1}^{L} k_i))$
Cartesian product	PQ	$O(\frac{D}{m}\sum_{i=1}^{m} k_i)$	$O(\frac{D}{m}\sum_{i=1}^{m} k_i)$
Linear combination	CQ	$O(D\sum_{i=1}^{L} k_i)$	$O(D\sum_{i=1}^{L} k_i)$

D: dimension of the raw vector; P: partitioin factor of HKM; K: overall codebook size; k_i: size of the i^{th} sub-codebook; L: number of levels of RVQ; m: number of sub-codebooks

Some hash-based methods, like spectral hashing (SH) (Weiss et al., 2008) and locality sensitive hashing (LSH) (Indyk and Motwani, 1998), have a projection stage which projects the vectors into the hash tables and is similar to the VQ process which projects vectors to their nearest neighbors in the codebook. However, these hash-based methods cannot be seen as VQ methods as none of them has a codebook which is a necessary requirement for VQ.

4.1 Tree-structured VQ

Tree-searched VQ (TSVQ) was first proposed by Buzo et al. (1980) for speech coding, and it outperforms the optimal scalar quantization method significantly with a given level of average distortion, achieving a lower data rate and a higher compression rate. In addition, the computational complexity is reduced by the tree search algorithm compared with a full search algorithm. According to Buzo et al. (1980), the tree is constructed by recursive splitting. The first level of the tree has n codewords if the branch number of the tree nodes is n and the vector space is split into n partitions. On the second level, there are n^2 codewords, and each partition of the upper level is split into n parts using the k-means algorithm. This procedure is repeated ℓ times if there are ℓ levels for the tree. In the end, n^ℓ codewords for ℓ levels are generated. Assuming that b is the number of bits used to encode each codeword, we have

$$n^\ell = 2^b \tag{11}$$

and

$$b = \ell \log_2 n. \tag{12}$$

When quantizing the vector, the algorithm finds the nearest codeword on each level and descends the tree node corresponding to the codeword. As a result, just a small part of the codewords are searched and the computational complexity is reduced in contrast with a full search algorithm which just uses one codebook corresponding to a tree with one node which has 2^b codewords.

Despite the efficiency of TSVQ for speech coding, TSVQ is a locally optimal algorithm for finding the nearest neighbor at each level. However, the authors of a recently proposed similar method called hierarchical k-means (HKM) (Nister and Stewenius, 2006) found it scalable to use a vocabulary tree for object recognition. Fig. 1 shows the structure of the HKM vocabulary tree. The automatic configuration of HKM according to the dataset has been implemented by FLANN (Muja and Lowe, 2009).

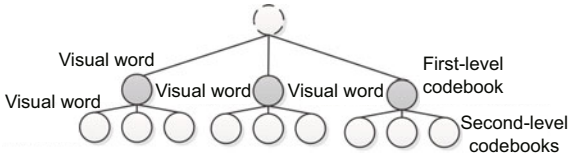

Fig. 1 Vocabulary tree of hierarchical k-means

Random projection tree (RPT) (Dasgupta and Freund, 2008, 2009) is a kind of TSVQ and is based on the observation that many high-dimensional data have low intrinsic dimension, which means that the data lie near a smooth low-dimensional manifold (Dasgupta and Freund, 2009). RPT seeks to automatically adapt to the intrinsic low-dimensional structure rather than explicitly learn it. To solve this problem, RPT exploits a hierarchical decomposition algorithm for the high-dimensional input data space and splits the space recursively until a desired quantization error is reached. The key lies in the splitting method, which picks a random direction from the surface of the unit sphere in $\mathbb{R}^D$ (D is the dimension) to split a region into two. Dasgupta and Freund (2009) offered two other splitting alternatives, namely the PCA-RPT which splits the region along the principal component direction, and the approximate PCA-RPT which computes the approximate principal eigenvectors by a stochastic gradient descent algorithm to reduce the computation cost of PCA-RPT. As a result, RPT is computationally very efficient and the quantization error depends only on the small intrinsic dimension rather than the large apparent dimension, alleviating the curse of dimensionality problem (Beyer et al., 1999).

The encoding process of TSVQ is very fast, and the encoding time complexity is proportional to the depth of the tree. TSVQ can obtain a large vocabulary from a small tree depth. Despite the efficiency of TSVQ, TSVQ is a locally optimal algorithm by finding the nearest neighbor at each level, which limits its accuracy.

4.2 Direct sum VQ

A direct sum VQ method can be defined as a method whose overall codebook is a direct sum of multiple sub-codebooks and the overall quantization result is the sum of quantization results of all the sub-quantizers.

4.2.1 Multiple stage VQ

Multiple stage VQ (MSVQ) was proposed by Juang and Gray (1982) for speech coding. MSVQ is a special case of TSVQ, with one node (or one codebook) at each level. To reduce computational and storage requirements, MSVQ employs several stage codebooks for quantization rather than one large codebook and the quantization is performed sequentially, namely stage by stage, and then the quantization result at each stage is merged to represent the input vector.

As described in Fig. 2, when learning the codebook, MSVQ obtains the codebook of the first stage by k-means, and the difference (or residual) between the raw input vector and its corresponding centroid is sent to the second stage, and the quantizer of the second stage generates the stage codebook by k-means just as in the first stage. Namely, the k^{th} stage (except the first stage) quantizer learns the k^{th} stage codebook on the residuals of the $(k-1)^{\text{th}}$ stage, so MSVQ is called residual VQ (RVQ).

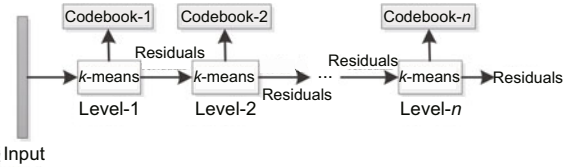

Fig. 2 Training process of multiple stage vector quantization

When quantizing the input vector, the quantizer first finds the nearest neighbor in the first-stage codebook, and then the difference between the input vector and the nearest centroid is sent to the next stage to find the nearest centroid in the second-stage codebook. Let $\boldsymbol{q}_j$ be the quantization result, namely the nearest centroid, of the j^{th} stage. Then the overall quantization result of MSVQ is

$$\hat{\boldsymbol{q}} = \sum_{j=1}^{L} \boldsymbol{q}_j, \tag{13}$$

and the overall codebook $\boldsymbol{C}$ is

$$\boldsymbol{C} = \boldsymbol{C}_1 + \boldsymbol{C}_2 + \cdots + \boldsymbol{C}_L, \tag{14}$$

where L is the number of stages and $\boldsymbol{C}_i$ is the codebook of the i^{th} stage. Eq. (14) means that the overall codeword of $\boldsymbol{C}$ is the sum of the stage codewords of all the stages. That is to say, the overall codebook is the direct sum of the stage codebooks. While the number of elements of vector set $\boldsymbol{C}$ may be large if L and the size of $\boldsymbol{C}_i$ are not small, the quantizer does not need to store $\boldsymbol{C}$; instead, just stage codebooks $\boldsymbol{C}_i$ are stored, and a lot of storage can be saved.

Let $\boldsymbol{q}_k$ of Eq. (13) be the i_k^{th} centroid of $\boldsymbol{C}_k$. Then the overall encoding result of the quantizer is

$$i = (i_1, i_2, \ldots, i_L), \tag{15}$$

which is L-tuple and an element of index space I is

$$I = I_1 \times I_2 \times \cdots \times I_L, \tag{16}$$

where I_j is the index set of the j^{th} stage and "$\times$" denotes the Cartesian product. As can be seen from Eq. (16), the encoding result is a product code, so MSVQ is called the summation product code VQ where "summation" means the direct sum codebook. Let k_j be the number of centroids of the j^{th} stage. Then size of space I is

$$N = \prod_{j=1}^{L} k_j, \tag{17}$$

which is very large if neither L nor k_j is small. Therefore, a sequential search constraint is employed when the quantizer quantizes the input vectors. Instead of searching the whole space of I, the sequential search strategy saves a lot of computation time and reduces the complexity of the quantizer. Though computationally efficient, the sequential search strategy is the shortcoming of MSVQ, which results in a locally optimal solution. However, this problem can be solved by jointly constructing the stage codebooks to obtain a sub-optimal solution close to the global optimal solution by searching the whole space of $\boldsymbol{C}$, which was explored by Chan and Gersho (1990), Chan et al. (1992), and Barnes et al. (1996), and used by Ai et al. (2014) for enhanced RVQ (ERVQ). Motivated by optimized PQ (OPQ) (Ge et al., 2013, 2014) that aims to find an optimal rotation matrix to reduce the quantization error, Guo et al. (2016) proposed non-parametric RVQ (RVQ-NP) and parametric RVQ (RVQ-P) to further improve the performance of RVQ.

However, when the dataset for learning the codebook is large and high-dimensional, MSVQ which uses k-means for many stages will be very slow, so some researchers seek to speed it up. Wei et al. (2014) proposed a variant of MSVQ called projected RVQ (PRVQ), which uses principal component analysis (PCA) to reduce the dimensionality and optimizes the projection matrix and codebook jointly. The idea is similar to iterative quantization (ITQ) (Gong and Lazebnik, 2011) and transform coding (TC) (Brandt, 2010), which make use of orthogonal transform. Similar to PRVQ, Liu et al. (2017) proposed generalized RVQ (GRVQ), which employs PCA and makes use of a transition clustering method to improve over k-means for high intrinsic dimensional data. An aggregating tree (A-Tree), which stores the encoding result in a radix tree to speed up search and reduce memory consumption, was proposed for large-scale ANN search. As a result, GRVQ outperforms state-of-the-art methods like AQ (Babenko and Lempitsky, 2014) in terms of both accuracy and search speed. Instead of generating a global rotation matrix for each layer, transformed residual quantization (TRQ) (Yuan and Liu, 2015b) generates an optimal rotation matrix for each cluster of each layer in that the shape and orientation of each cluster are different and a local rotation is needed rather than a global one.

4.2.2 Mean-removed VQ

Mean-removed VQ (MRVQ) or separating mean VQ (Gersho, 1979; Gray and Neuhoff, 1998; Jégou et al., 2010), in fact, is a direct sum VQ method. MRVQ is composed of two sub-quantizers, one for the mean of the vectors, which is a scalar quantizer,

and the other for the residuals, resulting in two codebooks (mean codebook and shape codebook).

For vector $\boldsymbol{x}$, the "mean" is the sample mean achieved by averaging all the dimensions of $\boldsymbol{x}$, i.e., $(x_1 + x_2 + \cdots + x_d)/d$, where d is the dimension of $\boldsymbol{x}$. When MRVQ learns the codebook, the mean codebook is achieved from the means of the input vectors and the shape codebook is achieved from the residuals between the input vectors and their corresponding means. When quantizing, MRVQ first obtains the mean of the input and then subtracts it from the input vector, resulting in the residual vector which is subsequently quantized by the second quantizer. As a result, the reproduction is just the sum of the quantization results of two quantizers. MRVQ achieves good reconstructed image quality, but it requires a high bit rate. To reduce the bit rate without much loss of the quality of the reconstructed image, Chuang et al. (2013) proposed improved MRVQ (IMRVQ), which first encodes the block mean values by a linear prediction technique. Then the Huffman coding algorithm is applied. The MRVQ indices of the residual vectors are further compressed by the Huffman coding algorithm. When the residual codewords are not suitable for encoding the image block, only the block mean value is used to encode it.

The motivation for MRVQ is that the "mean" represents the average background of an image and the "residual" represents the shape of the input vector about its mean. However, this meaning may be just proper for the vectors purely consisting of the pixels of an image. For the SIFT-like vectors which are constructed from the gradient histograms, the meaning may be different. As a result, this method is not widely used. The accuracy of direct sum VQ (or residual vector quantization) is usually very high. However, direct sum VQ uses a sequential search strategy, resulting in a locally optimal solution. Theoretically, the larger the number of stages, the smaller the quantization error. However, in fact, the number of stages of direct sum VQ cannot be too large in order to reduce the encoding time and memory consumption, especially when the vectors are high-dimensional.

4.3 Cartesian product VQ

Cartesian product VQ is a method which uses many sub-codebooks and the overall codebook is a Cartesian product of the sub-codebooks. Product VQ (PQ), Cartesian k-means (CKM), and transform coding (TC) belong to this category. Inverted multi-index (IMI), which just proposes a new index structure on PQ for ANN search, can be put into this category.

4.3.1 Product VQ

Product VQ was first proposed by Sabin and Gray (1982, 1984), then described in Gray and Neuhoff (1998), and finally detailed by Jégou et al. (2010) in theory and practice for an ANN search. The overall codebook of PQ is a Cartesian product of several low-dimensional codebooks. Just as in MSVQ, PQ was proposed to reduce the computational complexity and storage requirement of the quantizer.

For a product quantizer, a d-dimensional input vector is decomposed into m (a factor of d) parts or components and each part is $\frac{d}{m}$-dimensional just as follows:

$$\begin{aligned}\boldsymbol{V} = (&\underbrace{v_1, v_2, \ldots, v_k}_{\boldsymbol{u}_1}, \underbrace{v_{k+1}, v_{k+2}, \ldots, v_{2k}}_{\boldsymbol{u}_2},\\ &\ldots, \underbrace{v_{d-k+1}, v_{d-k+2}, \ldots, v_d}_{\boldsymbol{u}_m}),\end{aligned} \tag{18}$$

where $\boldsymbol{u}_j$ is the j^{th} sub-vector of $\boldsymbol{v}$. Let $q_j(\boldsymbol{x})$ be the j^{th} sub-quantizer for the j^{th} sub-vector. Then the overall quantizer $q(\boldsymbol{x})$ is

$$q(\boldsymbol{x}) = (q_1(\boldsymbol{u}_1), q_2(\boldsymbol{u}_2), \ldots, q_m(\boldsymbol{u}_m)). \tag{19}$$

Namely, the overall quantization result is a concatenation of the results of the sub-quantizers. Let $\boldsymbol{C}$ be the overall codebook and $\boldsymbol{C}_j$ the j^{th} codebook corresponding to $q_j(\boldsymbol{x})$, which can be achieved by the k-means algorithm on the j^{th} sub-vectors of all the input vectors used to learn the codebook, as is described in Fig. 3. Then we have

$$\boldsymbol{C} = \boldsymbol{C}_1 \times \boldsymbol{C}_2 \times \cdots \times \boldsymbol{C}_m. \tag{20}$$

As can be seen from Eq. (20), the overall codebook $\boldsymbol{C}$ is a Cartesian product of the sub-codebooks of the sub-quantizers. Let I_j be the index set corresponding to the j^{th} subquantizer. Then just as in the case of MSVQ, the overall index space is

$$I = I_1 \times I_2 \times \cdots \times I_m. \tag{21}$$

While the index structure of PQ is the same as that of MSVQ, the indices of PQ and MSVQ show

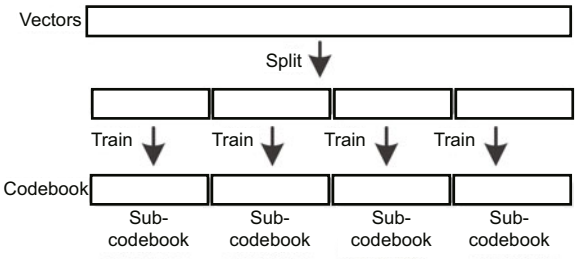

Fig. 3 Training process of product vector quantization (PQ)

different meanings. For MSVQ, I_j is the index set of the j^{th} stage, corresponding to the codebook generated on the residuals of the $(j-1)^{\text{th}}$ stage ($j>1$), while for PQ, I_j corresponds to the j^{th} codebook generated on the j^{th} sub-vectors.

Just like MSVQ, PQ does not need to store the overall codebook $\boldsymbol{C}$ and just needs to store sub-codebooks $\boldsymbol{C}_i$; as a result, storage space is saved. When quantizing, PQ quantizes the sub-vectors independently, which reduces the computation time in contrast to searching a large codebook $\boldsymbol{C}$ to obtain the nearest neighbor for quantization. Let k_j be the number of codewords of $\boldsymbol{C}_j$. Then the time needed for quantization is $O(\max(k_i)m)$ or $O(k_1+k_2+\cdots+k_m)$ instead of $O(k_1k_2\cdots k_m)$ or $O(\max(k_i)^m)$ for a full search of $\boldsymbol{C}$, which may be very large if m and k_j are not small.

As reported by Jégou et al. (2010), PQ outperforms two of the state-of-the-art methods for ANN search: spectral hashing (SH) (Weiss et al., 2008) and hamming embedding (HE) (Jégou et al., 2008). Though efficient and effective, PQ has one main shortcoming: the sub-vectors of a vector are supposed to be independent, which cannot be true for real-world data like SIFT (Lowe, 1999, 2004) descriptors, since they are structured and built on concatenated orientation histograms. For PQ, the bins of a histogram may end up in different quantization groups (Jégou et al., 2010). Jégou et al. (2010) have proposed a grouping method to assure mutual independence for the sub-vectors. However, it is not perfect as it cannot guarantee that the grouping method is optimal. Therefore, some researchers have begun to find the best decomposition for PQ, which leads to the occurrence of optimized PQ (OPQ) proposed by Ge et al. (2014). OPQ aims to find the optimal decomposition of the vector space.

In essence, the purpose of OPQ is to find an optimal orthogonal matrix $\boldsymbol{R}$, which decides the dimensions assigned to each subspace. The problem can be defined as

$$\begin{aligned}\min_{\boldsymbol{R},\boldsymbol{C}_1,\boldsymbol{C}_2,\ldots,\boldsymbol{C}_m}\ &\sum_{\boldsymbol{x}}\|\boldsymbol{R}\boldsymbol{x}-\boldsymbol{C}\|_{\mathrm{F}}^2\\ \text{s.t.}\quad &\boldsymbol{C}=\{\boldsymbol{c}|\boldsymbol{c}\in\boldsymbol{C}_1\times\boldsymbol{C}_2\times\cdots\times\boldsymbol{C}_m\},\\ &\boldsymbol{R}^{\mathrm{T}}\boldsymbol{R}=\boldsymbol{I},\end{aligned}\tag{22}$$

where $\boldsymbol{C}_j$ represents the sub-codebook of the j^{th} sub-vectors and $\boldsymbol{R}$ the orthogonal matrix that needs to be achieved by OPQ.

Ge et al. (2013, 2014) proposed two methods to solve $\boldsymbol{R}$ and $\boldsymbol{C}$: non-parametric method (OPQ-NP) and parametric method (OPQ-P). The non-parametric one uses the parametric method for initialization and achieves a higher accuracy. The non-parametric method, which does not assume a probability distribution for the input, first initializes $\boldsymbol{R}$ and $\boldsymbol{C}$ with the parametric method, then fixes $\boldsymbol{R}$ to optimize $\boldsymbol{C}$, and finally fixes $\boldsymbol{C}$ to optimize $\boldsymbol{R}$, leading to the orthogonal Procrustes problem (Schönemann, 1966) which is used by iterative quantization (ITQ) (Gong and Lazebnik, 2011). The parametric method assumes that each dimension of the input vectors follows a Gaussian distribution with zero mean. To achieve minimal quantization distortion, two criteria should be satisfied: (1) the mutual independence of dimensions, which can be achieved by aligning the data by PCA; (2) balanced subspaces' variances, which can be achieved by a procedure called "eigenvalue allocation." As a result, OPQ outperforms three state-of-the-art methods for ANN search: PQ, transform coding (TC) (Brandt, 2010), and iterative quantization (ITQ), with the same code length on publicly available datasets SIFT1M, GIST1M, and MNIST, respectively.

While OPQ is very good, some researchers observed that the performance of OPQ may be degraded for a strongly multi-modal distribution while the distribution of the within-cell (Voronoi cell) data is largely uni-modal. So, Kalantidi and Avrithis (2014) proposed the locally optimized product quantization (LOPQ) method, which learns a product quantizer per Voronoi cell on the residual distribution because the residuals are much more uni-modal than the original data. Like PQ, LOPQ has two quantizers: coarse quantizer $q_{\mathrm{c}}(\boldsymbol{x})$ and fine quantizer $q_{\mathrm{p}}(\boldsymbol{x})$. The coarse quantizer is an ordinary k-means quantizer for the original data, which is used to build the inverted lists, while the fine quantizer is

a product quantizer for the residuals, which is used to encode the residuals mapped into the inverted lists and each Voronoi cell (corresponding to one inverted list) will have one fine quantizer. When quantizing, LOPQ first uses $q_{\mathrm{c}}(\boldsymbol{x})$ to insert the input vector into one inverted list as follows:

$$L_i = \{j | E(\boldsymbol{x}_j) = i\}, \tag{23}$$

where $E(\boldsymbol{x})$ is the encoder which maps $\boldsymbol{x}$ to the partition number to which $\boldsymbol{x}$ is assigned, $\boldsymbol{x}_j$ represents the j^{th} input vector, and L_i is the i^{th} inverted list corresponding to the i^{th} partition generated by the coarse quantizer $q_{\mathrm{c}}(\boldsymbol{x})$. After the Voronoi cells of the partition or the inverted lists are generated, OPQ is applied locally for each cell; namely, the residuals of each cell are rotated locally by an orthogonal matrix, which can be described as

$$\begin{aligned} \boldsymbol{Z}_j &= \{\boldsymbol{x} - \boldsymbol{c}_j : q_{\mathrm{c}}(\boldsymbol{x}) = \boldsymbol{c}_j\} \\ &\rightarrow \hat{\boldsymbol{Z}}_j = \{\boldsymbol{R}_j^{\mathrm{T}} \boldsymbol{z} | \boldsymbol{z} \in \boldsymbol{Z}_j\} \\ &\rightarrow c_i^j = q_{\mathrm{p}}(\hat{\boldsymbol{z}}),\ \hat{\boldsymbol{z}} \in \hat{\boldsymbol{Z}}_j, \end{aligned} \tag{24}$$

where $\boldsymbol{c}_j$ is the j^{th} codeword of the coarse quantizer codebook, $\boldsymbol{R}_j$ the rotation matrix corresponding to the j^{th} Voronoi cell of the coarse quantizer, and c_i^j the i^{th} codeword of the j^{th} codebook corresponding to the j^{th} Voronoi cell of the coarse quantizer. As a result, LOPQ is very effective and outperforms many state-of-the-art methods for ANN search: Ck-means (Norouzi and Fleet, 2013), IVFADC of PQ, multi-D-ADC of the inverted multi-index method (Babenko and Lempitsky, 2012), and joint-ADC of the joint inverted index method (Xia et al., 2013) for recall@R criteria on SIFT1M, GIST1M, and the large-scale dataset SIFT1B. Rather than finding the optimal rotation matrix to decorrelate the subspaces of PQ, Yuan and Liu (2015a) proposed product tree quantization (PTQ) to combine HKM (Nister and Stewenius, 2006) with PQ. PTQ performs a two-level HKM on each subspace and this structure relaxes the independence constraint on the subspaces. What is more, an optimized PTQ (OPTQ) was proposed to combine a rotation matrix with PTQ just like OPQ. As the experiments of Yuan and Liu (2015a) showed, OPTQ achieves a higher accuracy than OPQ with an acceptable loss of speed.

4.3.2 Cartesian k-means (CKM)

CKM (Norouzi and Fleet, 2013) is an equivalent of OPQ (Ge et al., 2013), which uses k-means to generate the codebook, and the quantization error is closely related to the codebook size determined by k. However, as a result of storage and run-time requirement, k (number of codewords) cannot be too large, like 2^{64}. CKM finds a family of models to reduce the encoding cost of the k codewords from $O(k)$ to $O(\log_2 k)$ by a compositional parameterization of the codewords. Two such models have been proposed by Norouzi and Fleet (2013), namely orthogonal k-means (Ok-means), which is a generalization of iterative quantization (ITQ) (Gong and Lazebnik, 2011), and Cartesian k-means (Ck-means), which is a generalization of Ok-means and can be viewed as a generalization of PQ.

Ok-means represents each cluster center with an additive combination of the columns of matrix $\boldsymbol{C}$ which has m columns, and then there are 2^m possible combinations. As a result, the space of cluster centers is 2^m, meaning that k is 2^m. However, we just need to store $\boldsymbol{C}$ which just has $m = \log_2 k$ column vectors; consequently, the storage is saved. Then the key of Ok-means is to achieve the best $\boldsymbol{C}$ to minimize the quantization error. The authors restricted $\boldsymbol{C}$ to be orthogonal to use the result of the orthogonal Procrustes problem. Ck-means divides $\boldsymbol{C}$ into p parts or subsets, each with h vectors or columns and $m = ph$. Each cluster center is an additive combination of the vectors from p subsets, one vector per subset. Then the size of the cluster center space is $k = h^p$; each cluster center is a Cartesian product of the p subsets. To solve $\boldsymbol{C}$ easily, the authors assumed that the vectors in different subsets are pairwise orthogonal.

Experiments showed that CKM outperforms PQ and ITQ in accuracy for nearest neighbor search with the same k or encoding bits. However, both PQ and CKM just select one codeword from the subspace with each subspace just having one codebook; this may restrict the accuracy of ANN search. So, some researchers have sought to construct multiple codebooks for each subspace and select multiple sub-codewords from one subspace to represent the sub-vector of an input point. Optimized Cartesian k-means (OCKM) (Wang et al., 2014) is the one. By making use of this strategy, OCKM outperforms PQ and CKM as experiments suggest.

4.3.3 Transform coding (TC)

TC (Brandt, 2010) presented here is a generalization of the original transform coding for image

coding using VQ and operating on image descriptors like SIFT instead of pixels. TC combines the original transform coding (which allocates bits to the components of the vectors according to the data distribution) with product quantization. However, the product quantization used by TC is an extreme case of PQ using a scalar quantizer for each component to minimize the distortion. Therefore, each dimension of the vectors corresponds to a codebook and the key problem is how to decide each codebook's size corresponding to the bits allocated to each dimension. Since a dimension with larger variance carries more information, TC allocates bits to the dimensions according to the variance. The number of bits b_i allocated to the i^{th} dimension is

$$b_i = \log_2 \delta_i, \tag{25}$$

where δ_i is the standard deviation of the i^{th} dimension. However, some dimensions with small variance may be allocated to no bit, so PCA is applied to reduce these unimportant dimensions as well as statistical dependence among dimensions. An inverse transformation will be needed to map the low-dimensional quantized vector back into the original space. The whole process of TC can be described as

$$\begin{aligned} \boldsymbol{y} &= \mathrm{PCA}(\boldsymbol{x}) \\ &\rightarrow q(\boldsymbol{y}) = (q_1(y_1), q_2(y_2), \ldots, q_d(y_d)) \\ &\rightarrow \boldsymbol{x}' = \mathrm{PCA}^{-1}(q(\boldsymbol{y})). \end{aligned} \tag{26}$$

As a result, TC is very simple, fast, and effective, and can be further accelerated by graphics processing units. However, TC is not very accurate compared with PQ. TC assumes a uniform and identical distribution for each dimension after normalization, which cannot easily be satisfied by real-world data, so optimized TC (OTC) (Park et al., 2014), which optimizes the bit allocation process directly on the binned kernel estimator of each component of a vector, was proposed and reported to be nearly an order of magnitude faster than OPQ with nearly the same accuracy when using the same code length.

K-subspace quantization (KSQ) (Ozan et al., 2016a) is a variant of transform coding. KSQ has K affine subspaces corresponding to K clusters; each subspace has an affine shift vector and a rotation vector. The input vector is first projected to one of the K subspaces. Then scalar quantization is applied to each dimension of the projected vector. KSQ is different from TC, since the rotation of KSQ is performed locally (not globally) on clusters. Though simple, KSQ achieves comparable performance to state-of-the-art methods like additive quantization (AQ) (Babenko and Lempitsky, 2014) and composite quantization (CQ) (Zhang T et al., 2014).

4.3.4 Distance encoded VQ

To accelerate distance computation, Jégou et al. (2010) proposed two approximate distance computation methods: asymmetric distance computation (ADC) and symmetric distance computation (SDC). The nearest codeword was used to approximate the raw vector for distance computation. Distance encoded PQ (DPQ) (Heo et al., 2014) was proposed to address the distance approximation error by encoding both the quantization result and the distance. The distance space was partitioned into several bins by several thresholds. The Voronoi cell was further partitioned by the distance, much finer than PQ. What is more, two new distance metrics were proposed, namely, statistic- and geometry-based distance metrics. Combining the strategies of DPQ with OPQ (Ge et al., 2014), DPQ outperforms OPQ, especially for high-dimensional descriptors.

4.3.5 Inverted multi-index (IMI)

IMI (Babenko and Lempitsky, 2012) is the generalization of the inverted list indexing structure used in product quantization by replacing the standard VQ within inverted indices with PQ, meaning that the coarse quantizer (Jégou et al., 2010) used to build the inverted list is a product quantizer and that the inverted multi-index is a multi-dimensional table. As a result, a much finer division of the search space is achieved, and this leads to much shorter inverted lists and candidate lists. Consequently, IMI is much faster and more accurate for retrieval and approximate nearest neighbor search on large-scale datasets without much increasing memory overhead. However, the subdivision of the search space cannot be too fine, or there will be many empty lists and the quantization time will increase. In view of the trade-off between speed and accuracy, the coarse quantizer of IMI divides the space into two or four subspaces, which are called multi-D-ADC and multi-4-D-ADC, respectively. IMI can be combined with OPQ (Ge et al., 2014) or LOPQ (Kalantidi

and Avrithis, 2014) to further increase the accuracy, which is truly striking.

The advantage of Cartesian product VQ is that either the encoding process or the training process is very fast because of splitting the vectors and doing encoding and training on the sub-vectors. However, PQ assumes that each part is independent and an extra rotation is needed to gain the independence property. PQ is more suitable for the structured vector. In addition, the best number of parts (i.e., parameter m) is different for different kinds of vectors of different dimensions.

4.4 Lattice VQ (LVQ)

LVQ was first introduced by Gersho (1979), and some efficient coding algorithms have been developed for some special lattices like A_n ($n \geq 1$), D_n ($n \geq 2$), and E_n ($n = 6, 7, 8$) by Convay and Sloane (1982) for the regular structure of lattices. Consequently, LVQ is of importance in image coding (Zhang L et al., 2014). A lattice is a set of points defined by generator matrix $\boldsymbol{B}$ whose columns consist of linearly independent basis vectors (Eriksson and Agrell, 1996), which can be described as

$$\Lambda = \{\boldsymbol{Bu} | \boldsymbol{u} \in \mathbb{Z}^d\}, \tag{27}$$

where $\boldsymbol{u}$ is a d-dimensional integer vector. As can be seen from Eq. (27), lattice Λ is the linear combination of the column vectors of $\boldsymbol{B}$ with integer coefficients. Every lattice point is surrounded by a Voronoi region and the points are evenly distributed in space; as a result, the Voronoi regions form a tessellation. The codewords of a lattice quantizer are lattice points, resulting in a codebook with a regular structure.

Pyramid VQ (PVQ) (Fischer, 1986) is based on the cubic lattice points lying on the surface of an L-dimensional pyramid. PVQ was motivated by the geometric structure of the memory-less Laplacian source while the traditional LVQ is for memoryless uniform sources, and PVQ can be applied to an arbitrary vector dimension. PVQ uses ℓ_1 norm for error measure in contrast to the traditional ℓ_2 norm error measure, i.e., MSE. There is a product code PVQ, which quantizes the ℓ_1 norm of the vector with a scalar quantizer in addition to a pyramid vector quantizer for the normalized vector, which is a hybrid scalar-vector quantization method. Similar to the strategy of using ℓ_1 norm in PVQ, the ℓ_1-norm-based coding for LVQ (Patchoo et al., 2013) encodes the ℓ_1 norm of the vector instead of the vector itself, in combination with a bit-plane approach which is used to represent the vector. Just as in PVQ, the ℓ_1-norm-based coding can handle an arbitrary vector dimension. The bit-plane method can be efficiently implemented with adaptive arithmetic coding. ℓ_1-norm-based coding is very suitable for encoding transform, subband, and wavelet data (Patchoo et al., 2013).

LVQ is very efficient for uniform vector distribution. However, this is its drawback as it is difficult to satisfy this condition for real-world data. Tuytelaars and Schmid (2007) proposed a data-independent method with a regular lattice instead of a cluster method to construct the codebook on SIFT-like descriptors, and the data-independent method is fast and accurate for object recognition.

4.5 Classified VQ (CVQ)

CVQ (Ramamurthi and Gersho, 1986) was proposed to solve two main problems of VQ. One problem is edge degradation when conventional distortion measures such as MSE (which does not possess any edge preserving property) are used. The second problem is the high complexity of VQ, whose complexity increases exponentially with the bit rate and block size (the image is first partitioned into contiguous, non-overlapping, and square blocks to form the vectors). As a result, the edges are poorly coded and only small block sizes of 3×3 or 4×4 are experimentally feasible. In Fig. 4, CVQ divides the vectors into many classes. For each class, a codebook is constructed using the training vectors belonging to the corresponding class. The classifier can simply be k-means. In this case, CVQ is a two-level hierarchical k-means VQ. The distance measure of CVQ

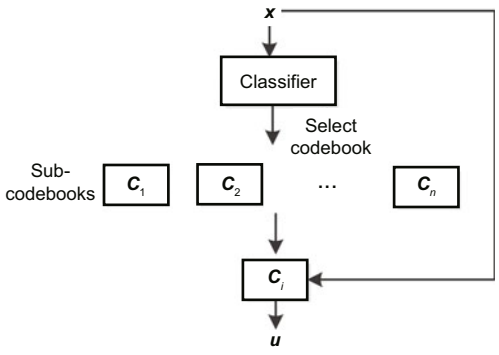

Fig. 4 Classified vector quantization

is different from the conventional Euclidean distance measure: when two vectors belong to the same class, the distance between them is the Euclidean distance; otherwise, the distance is infinite and the computational cost is reduced. Quadtree CVQ (QCVQ) (Chen et al., 2014), which uses quadtree segmentation to extract smooth and high-detailed regions of the image, is a modified version of CVQ. After quadtree segmentation, an edge-oriented classifier is employed to further classify the high-detailed regions into finer classes according to the edge orientations and textures. QCVQ can make use of both intra-block and inter-block correlation in images and achieve good performance for image retrieval.

CVQ can reduce the complexity of VQ by classifying the vectors into different classes. However, CVQ is not a common method, because it needs to train a classifier and operates on the raw pixels rather than vectors. As a result, the generalization ability of CVQ is poor.

4.6 Feedback VQ

Feedback VQ (Kieffer, 1982) is a kind of quantization method with memory whose current quantization result depends on its previous quantization result and is different from the memory-less quantization methods discussed previously. For a highly correlated source, feedback VQ is a promising method. A feedback vector quantizer consists of three components: an encoder that maps an input vector to a channel symbol or a channel codeword, a decoder that maps a channel symbol to a reproduction vector according to the current state, and a next-state function that produces the next state according to the current state and the channel symbol. The codebook of feedback VQ can be time-varying. Fig. 5 describes the encoding process of feedback VQ.

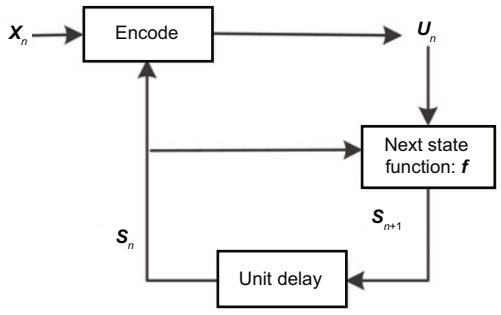

Fig. 5 Encoding process of the feedback vector quantization method

4.7 Fuzzy VQ

Finite state VQ (FSVQ) (Foster et al., 1985; Kim, 1988) is a kind of feedback VQ; a finite state vector quantizer is a finite state machine whose states are finite and can be used for speech and image coding. Predictive VQ (PreVQ) (Hang and Woods, 1985) is another feedback VQ method which is composed of a predictive part and a VQ part, and is a generalization of differential pulse code modulation (DPCM) which uses a scalar quantizer operating on a single pixel. PreVQ is more effective in removing the redundancy than using VQ.

Karayiannis and Pai (1995) proposed the fuzzy VQ (FVQ). It exploits the advantage of fuzzy clustering methods (Bezdek et al., 1984) to reduce the dependence of the resulting codebook on the random initial codebook selection (Karayiannis and Pai, 1995). Fuzzy clustering methods are based on soft decision which assigns a vector to multiple clusters. However, the VQ problem needs crisp or hard decision which assigns a vector to just one cluster, so some kind of strategy for the transition from soft to hard decision is needed. To solve this problem, FVQ allows assigning each vector to multiple clusters during the early stage of training, while hard decision is applied at the last stages of the clustering process. As stated in Karayiannis and Pai (1995), if the relative distortion rate decreases below a small threshold, all the training vectors are transferred to hard or crisp mode. Karayiannis and Pai (1995) proposed three FVQ methods: FVQ1, FVQ2, and FVQ3. The iteration procedure of the three methods is the same, while the membership function and computation of the codewords are different. The membership of FVQ1 is very simple, so it is faster than FVQ2 and FVQ3.

In contrast with fuzzy c-means (FCM) (Bezdek et al., 1984) whose central processing unit (CPU) time required by each iteration grows exponentially with k (the number of clusters), the average CPU time required by FVQ increases moderately and is computationally less demanding. Fuzzy learning VQ (FLVQ) (Bezdek et al., 1992), similar to FVQ, also employs FCM, but the difference is that FLVQ is a combination of FVQ and the Kohonen clustering network (KCN) (Kohonen et al., 1984), which is a self-organizing neural network. In other words, FLVQ is the Khonen network implementation of

FVQ. To overcome the drawbacks of KCN, FLVQ is non-sequential (independent on the order of the input), unsupervised, and uses fuzzy membership values from FCM as the learning rate. Though useful, the implementation of FLVQ has three main difficulties: (1) the selection of the initial value for the fuzziness parameter m; (2) high computational cost; (3) the transition from fuzzy mode (in which a training vector is assigned to multiple clusters) to crisp mode (in which a training vector is assigned to just one cluster). To solve these problems, Tsekouras et al. (2008) proposed the improved FLVQ (IFLVQ), which combines the benefit of FVQ and FLVQ and is (as the name implies) an improved version of FLVQ. In IFLVQ, a modified objective function of FCM is reformulated to manipulate the transition from fuzzy mode to crisp mode. IFLVQ gradually reduces the number of codewords to which a training vector is assigned, to accelerate the transition and reduce the computational cost. As a result, IFLVQ is fast and easy to implement.

Although very effective for the combination of c-means (or k-means) and fuzzy c-means, FVQ does not give a strict and precise mathematical explanation. Similar to IFLVQ, Tsekouras and Tsolakis (2013) proposed an improved version of FVQ, developing an objective function that incorporates the c-means and fuzzy c-means and providing a trade-off between speed and effectiveness. To further reduce the sensitivity of initialization, a codeword migration strategy, which migrates the codewords of the clusters with small utility value to positions close to clusters with large utility values, was proposed. The utility was used to measure the quality of the partition, and the utility of the i^{th} cluster, U_i, is defined as

$$U_i = \frac{D_i}{\sum_{j=1}^{k} D_j}, \tag{28}$$

$$D_i = \sum_{j=1}^{N} \mu_{ij} \|\boldsymbol{x}_j - \boldsymbol{c}_i\|_2^2, \tag{29}$$

where μ_{ij} is the membership function that controls the percentage of vector $\boldsymbol{x}_j$ belonging to cluster i. According to the utility measure, the fuzzy VQ method (Tsekouras and Tsolakis, 2013) performs better than other fuzzy methods like FVQ, FLVQ, and IFLVQ.

4.8 Linear combination VQ

Linear combination VQ can be defined as VQ methods which use multiple codebooks. A vector is approximated by a linear combination of codewords from the codebooks, one codeword for each codebook. These methods include AQ (Babenko and Lempitsky, 2014) and CQ (Zhang T et al., 2014). OCKM (Wang et al., 2014), which applies the linear combination strategy on each subspace, can be included. Linear combination VQ is similar to residual VQ. Both their global codebook structures are the direct sum of multiple sub-codebooks. However, the sub-codebooks of residual VQ are learned from residuals and are dependent on each other, while the sub-codebooks of linear combination VQ are mostly independently learned differently and usually some extra constraints are needed. Similar to RVQ, linear combination VQ does not have an independence assumption like PQ and assures a higher accuracy than PQ.

AQ (Babenko and Lempitsky, 2014), just like OCKM (Wang et al., 2014), is a similar method which uses the linear combination of M codewords from M different codebooks to approximate the vectors, but AQ is different from OCKM, because AQ does not divide the input space into several subspaces and does not make any subspace independence assumption, which is the drawback of PQ. Fig. 6 illustrates the encoding process. AQ finds one codeword from each sub-codebook to make the sum of these codewords as an approximate vector for the input vector. A beam search algorithm is used to solve the encoding problem to make the error between the input vector and the sum vector as small as possible. As a result, AQ is more accurate than PQ and OPQ

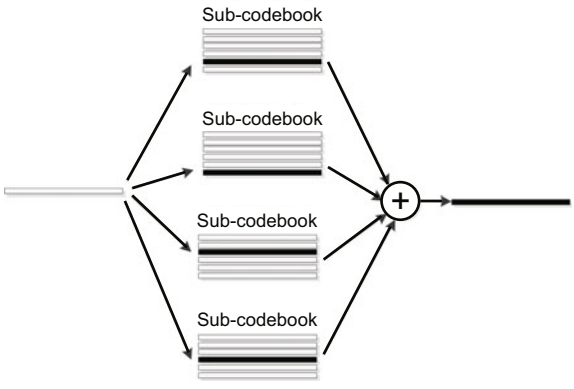

Fig. 6 Encoding process of additive quantization

with the same codebook size or the same candidate list length for ANN search. AQ allows fast ADC (Jégou et al., 2010) using the lookup table. The main drawback of AQ may be that learning the codebooks needs a lot of time for operating on the original input data rather than the sub-vectors and will consume much more memory. Thus, Babenko and Lempitsky (2014) proposed APQ that combines PQ and AQ and does AQ on the sub-vectors of PQ. APQ enjoys both the accuracy of AQ and the speed of PQ. Considering that the beam search that AQ uses for encoding is computationally expensive, Martinez et al. (2016) made use of stochastic local search for encoding and this method is called local search quantization (LSQ). The encoding process of LSQ is 30 to 50 times faster than that of the beam search of AQ, and the accuracy of LSQ outperforms those of the state-of-the-art CQ (Zhang T et al., 2014). To further speed up LSQ, a sparse version (SLSQ) was proposed, which imposes a sparsity constraint on the codebooks.

Tree quantization (TQ) (Babenko and Lempitsky, 2015) is a tree-structured AQ, which is a hybrid structure of TSVQ and AQ. The vertices correspond to the codebooks and the dimensions are assigned to the edges, one dimension for an edge. Each codebook encodes the dimensions that are assigned to edges incident to the vertex corresponding to this codebook (Babenko and Lempitsky, 2015). The dimensions that are not encoded by the codebook are fixed to zeros for the codewords. The codebooks that are not adjacent are orthogonal. The codebook learning process of TQ finally turns out an integer linear programming (ILP) problem and can be solved efficiently. Motivated by OPQ, a global rotation is performed, resulting in the optimized TQ (OTQ). The encoding procedure of OTQ is one order to two orders of magnitude faster than AQ. OTQ outperforms OPQ and AQ and is comparable to CQ (Zhang T et al., 2014) in accuracy. CQ is a similar method to AQ, but CQ applies an inter-dictionary-element-product constraint to accurately approximate the distance. CQ is more accurate than ITQ and CKM for ANN search without loss of speed. However, the search speed is seriously impeded by the distance lookup table computation procedure. Sparse CQ (SCQ) (Zhang et al., 2015) imposes a sparsity constraint on the codebooks to make use of sparse vector multiplication to accelerate distance lookup table construction. SCQ achieves the speed of PQ with little loss of accuracy.

Competitive quantization (CompQ) (Ozan et al., 2016b), motivated by the codebook joint optimization method of ERVQ (Ai et al., 2014), employs joint codebook optimization in CQ. Different from the optimization procedure of ERVQ, CompQ makes use of competitive learning (Ahalt et al., 1990) and stochastic gradient descent (SGD) to jointly learn all the codebooks. Consequently, CompQ outperforms CQ (Zhang T et al., 2014), OTQ (Babenko and Lempitsky, 2015), and KSQ (Ozan et al., 2016a), on SIFT1M and GIST1M.

Linear combination VQ is very accurate for ANN search, but the encoding process of most linear combination VQ is slower than that of PQ since linear combination VQ operates on the raw vector and does not split the vector like PQ. The memory consumption of the linear combination codebook is several times larger than that of PQ with the same number of codebooks.

4.9 Multi-quantizer VQ

According to the number of quantizers used, VQ methods can be divided into two categories: single- and multi-quantizer methods. The multi-quantizer VQ method uses multiple overall quantizers for quantization, not including methods with a coarse quantizer for building the index structure, like inverted multi-index, or the method which uses multiple sub-quantizers while the overall quantization result is the Cartesian product or the direct sum of the sub-quantizers, like Cartesian k-means, PQ, and RVQ. The multi-quantizer VQ method was specially proposed for ANN search. Joint inverted index (JII) (Xia et al., 2013) belongs to this category.

JII (Xia et al., 2013) is a multi-quantizer method; in essence, JII is just like locality sensitive hashing (LSH) (Indyk and Motwani, 1998) which uses multiple hash projection functions to project a point to multiple hash tables, and randomized k-d-tree (RKD-tree) (Silpa-Anan and Hartley, 2008), which uses multiple k-d trees for branch-and-bound search for ANN search. However, LSH and RKD-tree cannot be seen as VQ methods, because neither of them has a codebook to improve recall. JII uses multiple quantizers and optimizes all the codewords of all the quantizers jointly instead of optimizing each quantizer independently to achieve a globally

optimal solution that minimizes the total distortion of all the quantizers.

Assuming that JII has L quantizers and each quantizer has K codewords, then the algorithm for generating the codebooks of the quantizers for JII can be divided into three steps: (1) generate LK codewords by a single pass of k-means; (2) divide the LK codewords into K groups by a random projection tree (RP-tree/RPT) (Freund et al., 2007); (3) randomly select one codeword from each group for each quantizer without replacement. Thus, a K-codeword quantizer comes into being. Repeating the third step L times, L quantizers are formed. Each quantizer of JII is a simple VQ quantizer and each vector will have multiple approximate vectors. Like LSH, each quantizer of JII corresponds to an inverted index and a vector is assigned multiple times.

As shown in Xia et al. (2013), the joint quantizer generation method is effective and JII is more accurate than RKD, LSH, and k-means LSH (Paulevé et al., 2010) for ANN search. However, the superiority of JII is achieved at the cost of memory consumption. The memory cost of JII is several times larger than that of PQ.

5 Conclusions and future work

In this paper, we have reviewed the early proposed methods and their development for VQ and the recently proposed methods, especially for fast quantization-based ANN search application on large-scale high-dimensional descriptor datasets extracted from images.

5.1 Conclusions

The early methods were proposed for speech coding operating on speech sequences and image coding on image pixels; the recently proposed methods were proposed mainly for ANN search in the memory for large-scale image datasets. The key difference between these VQ methods is the codebook structure. There are mainly six codebook structures, tree (TSVQ), direct sum (RVQ), Cartesian product (PQ), lattice (LVQ), feedback (feedback VQ), and classified (CVQ), and two new codebook structures, linear combination (of many sub-codebooks) and joint (using multiple codebooks for multiple quantizers and optimizing them jointly). For ANN search application, four VQ methods have been extensively studied, i.e., tree-structured VQ, Cartesian product VQ, direct sum VQ, and linear combination VQ. The codebook structure of fuzzy VQ is very common and the difference is the codebook construction procedure which employs the fuzzy clustering approach. In fact, one more codebook structure may be included, that is, the hybrid codebook of hybrid VQ which combines different VQ approaches, such as TQ (Babenko and Lempitsky, 2015).

5.2 Challenges

Despite the tremendous progress in VQ, especially for ANN search applications, several major issues still need to be stressed:

1. The tree codebook can be very large to increase the accuracy without loss of the speed for searching just a part of the branches of the tree; of course, this consumes much memory to store the codebook tree.

2. The direct sum VQ and linear combination VQ are not practical for high-dimensional large-scale datasets performing on the raw vectors. The direct sum codebook and Cartesian product codebook are similar to the same index structure, and the search time and memory consumption are the sum of all the sub-quantizers. However, a method with a Cartesian product codebook is much faster and consumes less memory (by operating in the subspace) in contrast with a direct sum method which operates on the raw high-dimensional vectors, but is less accurate. The methods with a direct sum codebook or a Cartesian product codebook are truly striking for a relatively small codebook, fast speed, high accuracy, and are useful and promising for practical usage and future theory research.

3. A method with a lattice codebook is very fast for the regular structure of the codebook, but is not easy to generalize for the uniform distribution assumption.

4. The feedback codebook is not very common because of its complex structure and time-varying property.

5. The multi-quantizer has not been sufficiently studied because of its relatively large storage cost. The methods with a linear combination codebook or a joint codebook are multi-codebook methods, but different, as the method with a linear combination codebook is a single-quantizer method and the method with a joint codebook is a multi-quantizer

(with multiple overall quantizers) method.

5.3 Future directions

The recently proposed methods for approximate nearest neighbor search discussed in the study are mainly just a generalized and improved version of the early methods, but are applied mainly to descriptors extracted from images and used for ANN search. The improvement strategies of these methods suggest good ways to deal with the challenges of VQ in the future. Specifically, transform coding (Brandt, 2010) is a combination of the original transform coding and product quantization; inverted multi-index (Babenko and Lempitsky, 2012) has improved the indexing structure of PQ (Jégou et al., 2010) for ANN search and Cartesian k-means (Norouzi and Fleet, 2013). To a large extent, it is just an improved version of PQ and an equivalent of OPQ (Ge et al., 2013) for finding the optimal rotation matrix. LOPQ (Kalantidi and Avrithis, 2014) just applies the optimal rotation to each cluster. ERVQ (Ai et al., 2014) is an enhanced version of RVQ (Juang and Gray, 1982) for applying a joint optimization strategy to the codebooks. SCQ (Zhang et al., 2015) is a sparsity version of CQ (Zhang T et al., 2014) to improve speed using sparse matrix computation. The most different method is JII (Xia et al., 2013), since it uses multiple overall quantizers with a joint optimization method. As reported in Xia et al. (2013), JII is more accurate, but consumes much more memory. However, it can be a future research direction to develop quantization methods with multiple quantizers and comparable query speed, consuming less memory than JII.

Linear combination methods, with multiple subcodebooks and a proper optimization algorithm, have been the state-of-the-art methods. However, linear combination VQ performs on the raw vectors and the speed will be greatly influenced for high-dimensional descriptors with thousands of dimensions, like BOW (Sivic and Zisserman, 2003) and VLAD (Jégou et al., 2012). An efficient indexing structure will be a choice to solve this problem (Ai et al., 2013). The hybrid structures like PTQ (Yuan and Liu, 2015a) (a hybrid of PQ and TSVQ), APQ (Babenko and Lempitsky, 2014) (a hybrid of PQ and AQ), and TQ (Babenko and Lempitsky, 2015) (a hybrid of TSVQ and AQ), achieved excellent performance with regard to both speed and accuracy, and are becoming a promising research direction. However, much work is needed to find an optimal trade-off among speed, accuracy, and memory consumption.

References

Ahalt SC, Krishnamurthy AK, Chen P, et al., 1990. Competitive learning algorithms for vector quantization. *Neur Netw*, 3(3):277-290. https://doi.org/10.1016/0893-6080(90)90071-R

Ai LF, Yu JQ, He YF, et al., 2013. High-dimensional indexing technologies for large-scale content-based image retrieval: a review. *J Zhejiang Univ-Sci C (Comput & Electron)*, 14(7):505-520. https://doi.org/10.1631/jzus.CIDE1304

Ai LF, Yu JQ, Guan T, et al., 2014. Efficient approximate nearest neighbor search by optimized residual vector quantization. 12th Int Workshop on Content-Based Multimedia Indexing, p.1-4. https://doi.org/10.1109/CBMI.2014.6849842

Babenko A, Lempitsky V, 2012. The inverted multi-index. IEEE Conf on Computer Vision and Pattern Recognition, p.3069-3076. https://doi.org/10.1109/CVPR.2012.6248038

Babenko A, Lempitsky V, 2014. Additive quantization for extreme vector compression. IEEE Conf on Computer Vision and Pattern Recognition, p.931-938. https://doi.org/10.1109/CVPR.2014.124

Babenko A, Lempitsky V, 2015. Tree quantization for large-scale similarity search and classification. IEEE Conf on Computer Vision and Pattern Recognition, p.4240-4248. https://doi.org/10.1109/CVPR.2015.7299052

Barnes CF, Rizvi S, Nasrabadi NM, 1996. Advances in residual vector quantization: a review. *IEEE Trans Image Process*, 5(2):226-262. https://doi.org/10.1109/83.480761

Beyer K, Goldstein J, Ramakrishnan R, et al., 1999. When is "nearest neighbor" meaningful? In: Beeri C, Buneman P (Eds.), Database Theory—ICDT'99. Springer Berlin Heidelberg, p.217-235. https://doi.org/10.1007/3-540-49257-7_15

Bezdek JC, Ehrlich R, Full W, 1984. FCM: the fuzzy c-means clustering algorithm. *Comput Geosci*, 10(2-3):191-203. https://doi.org/10.1016/0098-3004(84)90020-7

Bezdek JC, Tsao ECK, Pal NR, 1992. Fuzzy kohonen clustering networks. IEEE Int Conf on Fuzzy Systems, p.1035-1043. https://doi.org/10.1109/FUZZY.1992.258797

Brandt J, 2010. Transform coding for fast approximate nearest neighbor search in high dimensions. IEEE Conf on Computer Vision and Pattern Recognition, p.1815-1822. https://doi.org/10.1109/CVPR.2010.5539852

Buzo A, Gray A, Gray R, et al., 1980. Speech coding based upon vector quantization. *IEEE Trans Acoust Speech Signal Process*, 28(5):562-574. https://doi.org/10.1109/TASSP.1980.1163445

Chan WY, Gersho A, 1990. Enhanced multistage vector quantization with constrained storage. 24th Asilomar Conf on Signals, Systems, and Computers, p.659-663. https://doi.org/10.1109/ACSSC.1990.523420

Chan WY, Gupta S, Gersho A, 1992. Enhanced multistage vector quantization by joint codebook design. *IEEE Trans Commun*, 40(11):1693-1697. https://doi.org/10.1109/26.179932

Chen HH, Ding JJ, Sheu HT, 2014. Image retrieval based on quadtree classified vector quantization. *Multim Tools Appl*, 72(2):1961-1984. https://doi.org/10.1007/s11042-013-1492-y

Chuang JC, Hu YC, Lo CC, et al., 2013. Improved mean-removed vector quantization scheme for grayscale image coding. *Int J Signal Process Image Process Patt Recogn*, 6(5):315-332. https://doi.org/10.14257/ijsip.2013.6.5.28

Convay JH, Sloane NJA, 1982. Fast quantizing and decoding algorithms for lattice quantizers and codes. *IEEE Trans Inform Theory*, 28(2):227-232. https://doi.org/10.1109/TIT.1982.1056484

Dasgupta S, Freund Y, 2008. Random projection trees and low-dimensional manifolds. 40th Annual ACM Symp on Theory of Computing, p.537-546. https://doi.org/10.1145/1374376.1374452

Dasgupta S, Freund Y, 2009. Random projection trees for vector quantization. *IEEE Trans Inform Theory*, 55(7):3229-3242. https://doi.org/10.1109/TIT.2009.2021326

Eriksson T, Agrell E, 1996. Lattice-Based Quantization: Part II. Technical Report No. 18, Chalmers University of Technology, Göteborg, Sweden.

Fischer T, 1986. A pyramid vector quantizer. *IEEE Trans Inform Theory*, 32(4):568-583. https://doi.org/10.1109/TIT.1986.1057198

Foster J, Gray R, Dunham M, 1985. Finite-state vector quantization for waveform coding. *IEEE Trans Inform Theory*, 31(3):348-359. https://doi.org/10.1109/TIT.1985.1057035

Freund Y, Dasgupta S, Kabra M, et al., 2007. Learning the structure of manifolds using random projections. 20th Int Conf on Neural Information Processing Systems, p.473-480.

Ge TZ, He KM, Ke QF, et al., 2013. Optimized product quantization for approximate nearest neighbor search. IEEE Conf on Computer Vision and Pattern Recognition, p.2946-2953. https://doi.org/10.1109/CVPR.2013.379

Ge TZ, He KM, Ke QF, et al., 2014. Optimized product quantization. *IEEE Trans Patt Anal Mach Intell*, 36(4):744-755. https://doi.org/10.1109/TPAMI.2013.240

Gersho A, 1979. Asymptotically optimal block quantization. *IEEE Trans Inform Theory*, 25(4):373-380. https://doi.org/10.1109/TIT.1979.1056067

Gersho A, 1982. On the structure of vector quantizers. *IEEE Trans Inform Theory*, 28(2):157-166. https://doi.org/10.1109/TIT.1982.1056457

Gersho A, Gray R, 1991. Vector Quantization and Signal Compression. Springer, Berlin, Germany.

Gong YC, Lazebnik S, 2011. Iterative quantization: a procrustean approach to learning binary codes. IEEE Conf on Computer Vision and Pattern Recognition, p.817-824. https://doi.org/10.1109/CVPR.2011.5995432

Gray R, 1984. Vector quantization. *IEEE ASSP Mag*, 1(2):4-29. https://doi.org/10.1109/MASSP.1984.1162229

Gray R, Neuhoff DL, 1998. Quantization. *IEEE Trans Inform Theory*, 44(6):2325-2383. https://doi.org/10.1109/18.720541

Guo D, Li CQ, Wu L, 2016. Parametric and nonparametric residual vector quantization optimizations for ANN search. *Neurocomputing*, 217:92-102. https://doi.org/10.1016/j.neucom.2016.04.061

Hang HM, Woods JW, 1985. Predictive vector quantization of images. *IEEE Trans Commun*, 33(11):1208-1219. https://doi.org/10.1109/TCOM.1985.1096238

Heo JP, Lin Z, Yoon SE, 2014. Distance encoded product quantization. IEEE Conf on Computer Vision and Pattern Recognition, p.2139-2146. https://doi.org/10.1109/CVPR.2014.274

Indyk P, Motwani R, 1998. Approximate nearest neighbors: towards removing the curse of dimensionality. 30th Annual ACM Symp on Theory of Computing, p.604-613. https://doi.org/10.1145/276698.276876

Jégou H, Douze M, Schmid C, 2008. Hamming embedding and weak geometric consistency for large-scale image search. 10th European Conf on Computer Vision, p.304-317. https://doi.org/10.1007/978-3-540-88682-2_24

Jégou H, Douze M, Schmid C, 2010. Product quantization for nearest neighbor search. *IEEE Trans Patt Anal Mach Intell*, 33(1):117-128. https://doi.org/10.1109/TPAMI.2010.57

Jégou H, Perronnin F, Douze M, et al., 2012. Aggregating local image descriptors into compact codes. *IEEE Trans Patt Anal Mach Intell*, 34(9):1704-1716. https://doi.org/10.1109/TPAMI.2011.235

Juang BH, Gray A, 1982. Multiple stage vector quantization for speech coding. IEEE Int Conf on Acoustics, Speech, and Signal Processing, p.597-600. https://doi.org/10.1109/ICASSP.1982.1171604

Kalantidi Y, Avrithis Y, 2014. Locally optimized product quantization for approximate nearest neighbor search. IEEE Conf on Computer Vision and Pattern Recognition, p.2329-2336. https://doi.org/10.1109/CVPR.2014.298

Karayiannis NB, Pai PI, 1995. Fuzzy vector quantization algorithms and their application in image compression. *IEEE Trans Image Process*, 4(9):1193-1201. https://doi.org/10.1109/83.413164

Kieffer J, 1982. Stochastic stability for feedback quantization schemes. *IEEE Trans Inform Theory*, 28(2):248-254. https://doi.org/10.1109/TIT.1982.1056487

Kim T, 1988. New finite state vector quantizers for images. Int Conf on Acoustics, Speech, and Signal Processing, p.1180-1183. https://doi.org/10.1109/ICASSP.1988.196809

Kohonen T, Huang T, Schroeder M, 1984. Self-Organization and Associative Memory. Springer-Verlag, Berlin, p.3406-3409. https://doi.org/10.1007/978-3-662-00784-6

Liu SC, Shao JR, Lu HT, 2017. Generalized residual vector quantization and aggregating tree for large-scale search. *IEEE Trans Multim*, 19(8):1785-1797. https://doi.org/10.1109/TMM.2017.2692181

Lloyd S, 1982. Least squares quantization in PCM. *IEEE Trans Inform Theory*, 28(2):129-137. https://doi.org/10.1109/TIT.1982.1056489

Lowe DG, 1999. Object recognition from local scale-invariant feature. 7th IEEE Int Conf on Computer Vision, p.1150-1157. https://doi.org/10.1109/ICCV.1999.790410

Lowe DG, 2004. Distinctive image features from scale-invariant keypoints. *Int J Comput Vis*, 60(2):91-110. https://doi.org/10.1023/b:visi.0000029664.99615.94

Martinez J, Clement J, Hoos HH, et al., 2016. Revisiting additive quantization. 14th European Conf on Computer Vision, p.137-153. https://doi.org/10.1007/978-3-319-46475-6_9

Muja M, Lowe DG, 2009. Fast approximate nearest neighbors with automatic algorithm configuration. 4th Int Conf on Computer Vision Theory and Applications, p.331-340. https://doi.org/10.5220/0001787803310340

Nister D, Stewenius H, 2006. Scalable recognition with a vocabulary tree. IEEE Conf on Computer Vision and Pattern Recognition, p.2161-2168. https://doi.org/10.1109/CVPR.2006.264

Norouzi M, Fleet DJ, 2013. Cartesian k-means. IEEE Conf on Computer Vision and Pattern Recognition, p.3017-3024. https://doi.org/10.1109/CVPR.2013.388

Oliva A, Torralba A, 2001. Modeling the shape of the scene: a holistic representation of the spatial envelope. *Int J Comput Vis*, 42(3):145-175. https://doi.org/10.1023/A:1011139631724

Ozan EC, Kiranyaz S, Gabbouj M, 2016a. K-subspaces quantization for approximate nearest neighbor search. *IEEE Trans Knowl Data Eng*, 28(7):1722-1733. https://doi.org/10.1109/TKDE.2016.2535287

Ozan EC, Kiranyaz S, Gabbouj M, 2016b. Competitive quantization for approximate nearest neighbor search. *IEEE Trans Knowl Data Eng*, 28(11):2884-2894. https://doi.org/10. 1109/TKDE.2016.2597834

Park M, Gunda K, Gupta H, et al., 2014. Optimized transform coding for approximate KNN search. British Machine Vision Conf, p.1-12. https://doi.org/10.5244/c.28.34

Patchoo W, Fischer TR, Maddex C, 2013. L_1-norm-based coding for lattice vector quantization. Asia-Pacific Signal and Information Association Annual Summit and Conf, p.1-4. https://doi.org/10.1109/APSIPA.2013.6694164

Paulevé L, Jégou H, Amsaleg L, 2010. Locality sensitive hashing: a comparison of hash function types and querying mechanisms. *Patt Recogn Lett*, 31(11):1348-1358. https://doi.org/10.1016/j.patrec.2010.04.004

Perronnin F, Dance C, 2007. Fisher kernels on visual vocabularies for image categorization. IEEE Conf on Computer Vision and Pattern Recognition, p.1-8. https://doi.org/10.1109/CVPR.2007.383266

Ramamurthi B, Gersho A, 1986. Classified vector quantization of images. *IEEE Trans Commun*, 34(11):1105-1115. https://doi.org/10.1109/TCOM.1986.1096468

Sabin M, Gray R, 1982. Product code vector quantizers for speech waveform coding. Global Telecommunications Conf, p.1087-1091. https://doi.org/10.1109/CVPR.2007.383266

Sabin M, Gray R, 1984. Product code vector quantizers for waveform and voice coding. *IEEE Trans Acoust Speech Signal Process*, 32(3):474-488. https://doi.org/10.1109/TASSP.1984.1164346

Schönemann P, 1966. A generalized solution of the orthogonal Procrustes problem. *Psychometrika*, 31(1):1-10. https://doi.org/10.1007/bf02289451

Shannon CE, 1948. A mathematical theory of communication. *Bell Syst Tech J*, 27(3):379-423. https://doi.org/10.1002/j.1538-7305.1948.tb01338.x

Shannon CE, 1959. Coding theorems for a discrete source with a fidelity criterion. IRE National Convention Record, Part 4, p.93-126. https://doi.org/10.1109/9780470544242.ch21

Silpa-Anan C, Hartley R, 2008. Optimized KD-trees for fast image descriptor matching. IEEE Conf on Computer Vision and Pattern Recognition, p.1-8. https://doi.org/10.1109/CVPR.2008.4587638

Sivic J, Zisserman A, 2003. Video Google: a text retrieval approach to object matching in videos. 9th IEEE Int Conf on Computer Vision, p.1470-1477. https://doi.org/10. 1109/ICCV.2003.1238663

Tsekouras GE, Tsolakis DM, 2013. Fuzzy clustering-based vector quantization for image compression. In: Chatterjee A, Siarry P (Eds.), Computational Intelligence in Image Processing. Springer Berlin Heidelberg, p.93-105. https://doi.org/10.1007/978-3-642-30621-1_5

Tsekouras GE, Antonios M, Anagnostopoulos C, et al., 2008. Improved batch fuzzy learning vector quantization for image compression. *Inform Sci*, 178(20):3895-3907. https://doi.org/10.1016/j.ins.2008.05.017

Tuytelaars T, Schmid C, 2007. Vector quantizing feature space with a regular lattice. IEEE 11th Int Conf on Computer Vision, p.1-8. https://doi.org/10.1109/ICCV.2007.4408924

Wang JF, Wang JD, Song JK, et al., 2014. Optimized Cartesian k-means. *IEEE Trans Knowl Data Eng*, 27(1):180-192. https://doi.org/10.1109/TKDE.2014.2324592

Wei BC, Guan T, Yu JQ, 2014. Projected residual vector quantization for ANN search. *IEEE Multim*, 21(3):41-51. https://doi.org/10.1109/MMUL.2013.65

Weiss Y, Torralba A, Fergus R, 2008. Spectral hashing. 21st Int Conf on Neural Information Processing Systems, p.1753-1760.

Xia Y, He KM, Wen F, et al., 2013. Joint inverted indexing. IEEE Int Conf on Computer Vision, p.3416-3423. https://doi.org/10.1109/ICCV.2013.424

Yuan JB, Liu XW, 2015a. Product tree quantization for approximate nearest neighbor search. IEEE Int Conf on Image Processing, p.2035-2039. https://doi.org/10.1109/ICIP.2015.7351158

Yuan JB, Liu XW, 2015b. Transformed residual quantization for approximate nearest neighbor search. arXiv:1512.06925

Zhang L, Zhang M, Pan Q, et al., 2014. An effective image coding method using lattice vector quantization in wavelet domain. *Int J Signal Process Image Process Patt Recogn*, 7(2):305-316. https://doi.org/10.14257/IJSIP.2014.7.2.28

Zhang T, Du C, Wang JD, 2014. Composite quantization for approximate nearest neighbor search. 31st Int Conf on Machine Learning, p.838-846.

Zhang T, Qi GJ, Tang JH, et al., 2015. Sparse composite quantization. IEEE Conf on Computer Vision and Pattern Recognition, p.4548-4556. https://doi.org/10.1109/CVPR.2015.7299085