

# Fast vector quantization based on subcodebook selection and its application to speech recognition

*José A. R. Fonollosa*

TALP Research Center  
Universitat Politècnica de Catalunya  
[www.talp.upc.es](http://www.talp.upc.es)

## Abstract

Vector quantization (VQ) is a efficient technique for data compression with a minimum distortion. VQ is widely used in applications as speech and image coding, speech recognition, and image retrieval. This paper presents a novel fast nearest-neighbor algorithm and shows its application to speech recognition. The proposed algorithm is based on a fast pre-selection that reduces the search to a limited number of code vectors. The presented results show that the computational cost of the VQ stage can be significantly reduced without affecting the performance of the speech recognizer.

## 1. Introduction

Vector quantization (VQ) is a powerful and popular method for data compression [1]. In the encoding process, the standard full-search VQ finds the nearest code vector for each input vector by exhaustively computing its distance to each of the code vectors in the codebook. Then, the index of the nearest neighbor is used to encode or represent the input vector.

In speech recognition systems based on Semi Continuous Hidden Markov Models (SCHMM) and in other applications, the search is extended to the  $k$ -nearest-neighbors.

Although VQ is an very efficient technique, its practical application is limited by the computational cost of the nearest-neighbor algorithm. This has motivated the research in the development of methods to accelerate the VQ process. We can classify previous fast algorithms into two groups.

The first group consider structured codebooks (tree-structured VQ, multi-stage VQs, lattice VQ, ...) that are sub-optimal because they impose constraints in the code vectors, the search pattern or in both. For example, in tree-structured VQ [1], the search is performed in stages. At each stage or node of the tree, the distance of the input vector to  $m$  representing vectors is computed. The nearest representing vector determines which of the  $m$  output paths is selected to reach the next node. This selection implies a reduction in the number of candidate vectors to  $1/m$ th the previous set. The result is a very fast algorithm, at the expenses of a sub-optimal result. The resulting code vector may not be the one with the lowest distance.

The second group of algorithms consider unstructured codebooks and the exact nearest-neighbor solution. Most of them exploits the inherent structure of the code vectors and offer very good results if the vector components are not independent [2]. Moreover, although the code vectors are not constrained to a structure, the algorithms are usually based on a search tree or a multistage approach.

The most effective algorithms begin with an initial structured search. This first step identifies a region in the space and the subset of code vector inside it. Then, a post-processing algorithm based on geometric properties is applied to find the optimal vector inside or around the cluster. In the next section, we present a short review of this kind of algorithms.

In this paper we propose an algorithm with a similar initial approach but with a different post-processing strategy. The proposed algorithm performs the search in two steps. The first step pre-selects a reduced number of code vectors using a low-cost method. Then, in the second step, the algorithm performs an exhaustive search restricted to this pre-selected cluster of code vectors.

In the proposed algorithm, the preselected subset of code vector indexes are precalculated off-line and stored in a table. To determine the table, we propose here a statistically-trained approach based on a training set of representing input vectors. The proposed search algorithm does not guarantee the exact  $k$ -nearest code vector indexes in the 100% of the cases. However, the performance on the tested scenario is as good as the obtained with an exact brute force search with a important reduction in the computational cost and a simple implementation.

Moreover, in most applications the codebook itself is usually designed using a training data and an algorithm that does not guarantee the optimal set of code vectors. Therefore, we can employ the same training data to design the look-up table of the proposed almost optimal fast algorithm without any practical degradation in performance.

A standard simple full-search improvement known as partial distance search algorithm (PDS) [3] is also considered in this paper for the second step. Other fast search algorithms briefly described in the following section may be as well combined with the proposed algorithm to accelerate the second step.

## 2. Fast nearest-neighbor algorithms

Given a codebook of  $N$  code vectors of dimension  $d$ , the brute force algorithm requires the computation of  $N$  distances. If we consider a Euclidean distance:

$$D^2(x, c) = \sum_{i=1}^d (x_i - c_i)^2 \quad (1)$$

the resulting algorithm requires a search time of  $O(Nd)$ .

To reduce the cost of this approach, many fast encoding algorithms have been proposed in the literature. Most of them are based on the combination of the following techniques.

### 2.1. Partial Distance Search

In the exhaustive-search approach, the distance between the input vector and each code vector is first computed using (1) and then compared to the distance to the  $k$ th nearest-neighbor found so far. A simple variation of this algorithm originally proposed by Cheng et al. [3] consists in performing the comparison inside the loop that computes the distance sum. If the partial sum exceeds the distance to the  $k$ th nearest-neighbor found so far, the rest of the loop is skipped.

Although the PDS algorithm increases the number of instructions inside the loop, the important reduction in the number of loop iterations give an overall substantial decrease in the computational cost. The PDS approach can be optimized by a principal component rotation of the codebook. Since this rotation causes the partial distance along the directions of largest variance to be calculated first in the loop, the loop iterations are further reduced [4].

### 2.2. Lower-bound based algorithms

Typically, these algorithms avoid the calculation of all the distances by the estimation of a lower bound. If the lower bound on the distance between the input vector and a code vector (or a set of code vectors) is greater than the distance to the  $k$ th nearest-neighbor found so far, the vector (or set of code vectors) is discarded without the need of computing the corresponding distance. An example for Euclidean distance can be found in [5] and a more general approach based on the triangle inequality in [6].

### 2.3. Projection methods

Fast nearest-neighbor algorithms based on projection try to translate the search from the  $d$ -dimensional space to a linear subspace. For example, the equal-average nearest-neighbor search uses the mean of the input vector (projection onto the central direction  $(1; 1; \dots; 1)$ ) to search and select a subset of the previously mean-ordered code vectors [7][8]. This simple method has received great attention and several extensions can be found in the literature. However, this approach gives good results only if the central direction is close to the principal component direction.

A more rigorous and general approach is to consider the projection directions given by a principal component analysis as proposed by C.Y. Chen et al. [9], and by S.C. Tai et al. [10], among others.

### 2.4. Search Trees

Search Trees is a usual approach to obtain fast search algorithms. Search trees divide the code vectors into two or more distinct subsets. Each subset is recursively subdivided until the number of code vectors in the terminal subset is small. The search for the nearest neighbors begins with the root node and work toward the terminal nodes. Sub-optimal tree-searched vector quantizers (TSVQ) were first proposed by Buzo et al [11] as a natural byproduct of the splitting algorithm for generating initial code guesses in the codebook design.

The TSVQ doubles the storage requirements, so its application to large codebooks is still limited. The Multi-stage

VQ originally proposed by B.-H. Juang and Gray [12] is a sub-optimal fast VQ with an important reduction on storage requirements. In the multi-stage VQ the search is also performed in two or more stages, but we have now only one set of the code vectors at each stage. The input vector is first quantized by the first codebook. Then, the difference between the input vector and the selected code vector is computed and successively applied as input to the following stages. Its practical application is limited by the constraints on the equivalent code vectors and by the sub-optimal multi-stage search procedure that severely limits its performance.

Search trees or multi-stage approaches can also be applied to unconstrained codebooks and exact fast nearest-neighbor searches. For example, an algorithm based on principal axis trees called PAT was proposed by McNames [2]. The PAT algorithm uses an efficient search tree to partition the code vectors into regions that have approximately the same density locally. A recursive binary search from the root node until a terminal node identifies the partition that contains the input vector. After computing the distance to code vectors in this partition, the algorithm consider other sibling partitions and applies elimination strategies using lower bounds.

A simpler algorithm for unconstrained codebooks based on a multi-stage approach was proposed by Woo et al. in [13]. The first stage of this algorithm uses a small codebook to identify a first coarse partition. In the second stage the search is restricted to the subset of code vectors which Voronoi partitions are inside or intersect with the first coarse partition. However, for large codebooks and large dimensions its practical interest is quite limited because the subset tends to be very large and may require a prohibitive time-consuming preprocessing.

### 2.5. Fine-coarse VQ

The search-tree or multi-stage approaches described above start with the identification of a coarse partition of the search space. The size of this partition is then successively reduced until the identification of the code vector, i.e., the associated Voronoi partition in which the input vector lies.

Moayeri proposed in [14] the opposite two-stage strategy called fine-coarse VQ (FCVQ). In the first stage, a fine quantizing partition is identified using a very large structured codebook. The output of this first fine quantizer, is then used as input by the unstructured codebook of interest. The benefit of FCVQ is that the second stage of quantization can be performed by table lookup, and consequently, does not add arithmetic complexity to that of the fast first stage. FCVQ is not an exact method, but its performance can be as close to the optimum as desired if sufficient storage for the fine structured quantizer and the lookup table is available.

A similar strategy named subcodebook searching (SCS) was proposed by Tai et al in [10]. Here, the first fine stage of SCS selects a subset of code vectors instead of a single code vector. Then, a full search is performed restricted to this preselected subset.

The new algorithm described in the next section, named subcodebook pre-selection (SCPS), extends the FSVQ, SCS and multi-stage [13] approaches to the  $k$ -nearest-neighbors case.

### 3. Subset selection based on the $k_1$ -nearest-neighbors of a smaller codebook

This section describes a new  $k$ -nearest-neighbors searching algorithm based on a first pre-selection that restricts the search to a limited subset of code vectors. This paper studies the selection of code vectors to be searched based on the  $k_1$ -nearest-neighbors of a smaller codebook.

#### 3.1. Search Algorithm

In the first stage the algorithm finds the  $k_1$  nearest code vectors of a small codebook. Then, the  $k_1$  indexes of this first stage are combined and used to address a lookup table that selects a reduced number of code vectors to be searched in the second stage. A standard  $k$ -nearest-neighbors full-search or PDS method is finally applied to the subset of selected code vectors.

#### 3.2. Codebook design

The codebook of the second stage is the unconstrained codebook of interest. It can be designed with any standard procedure using a training sequence. In our experiments the smaller codebook of the first stage was also designed with the same standard procedure and training sequence.

#### 3.3. Lookup table

The lookup table maps the  $k_1$  indexes of the first stage with the list of possible  $k$ -nearest-neighbors of the second stage. The combination of the  $k_1$  indexes of the first stage represents a fine partition of the search space. Therefore, for each combination of indexes, it is required to determine the Voronoi regions of the second stage that are inside or intersect the associated partition. An exhaustive search method can be used to construct a complete list of code vectors. This complete list of candidates would result in an algorithm completely equivalent to the full search case.

However, this exhaustive search is a time consuming process for creating the lookup table. Moreover, we are not interested here in a complete list, but in a 'representative' list with an equivalent performance and a minimal computational cost.

In the experiments presented in the next section the lookup table is constructed using a very large training sequence. For this purpose, the training sequence is quantized with the first and second codebook using an exact  $k$ -nearest-neighbors full-search algorithm. The complete list of the  $k$ -nearest indexes of the second stage is then associated with combined index formed with the  $k_1$  indexes of the first stage.

## 4. Results

The proposed algorithm was developed with the objective of reducing the computational cost of IberVox. IberVox is a widely used commercial product based on licensed speech technology of the Universidad Polit cnica de Catalu a and additional software developed by Applied Technologies on Language and Speech S.L. (www.atlas-cti.com). IberVox is based on semicontinuous hidden Markov models. The sampled signal is processed to obtain three value vectors of dimension  $d = 14$ : the MFCC (Mel-Frequency Cepstral Coefficients) and their two first temporal derivatives. Each

of these vectors is vector quantified to the closest 6 (from a total of 512) code vectors of the corresponding codebook.

For tasks with small vocabularies, as connected digit recognition, the computational cost of the vector quantizers represented the 30% of the total computational cost. The application of the algorithm described in this paper reduced in a 80% the computational load of the vector quantizer. For that recognition task, the total reduction of the computational cost of the system was a 25%. In order to compare this results with other fast algorithms we have to keep in mind that most of them are optimized or valid only for the single nearest-neighbor case. Very few publications has studied optimized algorithms or presented results for the  $k$ -nearest-neighbors case.

The codebooks and lookup tables were trained using the Spanish SpeechDat databases [15] with a total of more than 500.000 training vectors for each codebook. For the first stage of the proposed SCPS algorithm we considered two codebook sizes. A codebook of size  $N_1 = 64$  with  $k_1 = 2$  nearest indexes, and a codebook of size  $N_1 = 16$  with  $k_1 = 3$  nearest indexes. In both cases the lookup tables required about 250 Kbytes of extra memory.

For the creation of the lookup table we obtained the exact VQ results with the smaller codebook proposed for the first stage and with the second codebook of size  $N = 512$ . Then, we computed the probability of appearance of the indexes of the second codebook for a given combination of  $k_1$  indexes of the first codebook. The size of the table and the performance of the algorithm was optimized based on the observation that code vectors of very low probability (outliers) can be discarded from the lookup table without any loss in recognition performance.

In both stages of the SCPS algorithm we also consider the inclusion of the PDS method. Moreover, the effectiveness of PDS in the second stage was optimized by sorting the subcodebook indexes as a function of its probability given the  $k_1$  indexes of the first stage. PDS is more effective if, in average, we consider first the associated code vectors with higher probability.

We included the SCPS algorithm in the VQ module of IberVox to study its effect in different speech recognition tasks using testing signals outside the training data. In all the cases the performance of the recognizer was as good as with the full-search algorithm, with a significant reduction in computational cost. Table 1 shows the average number of computed distances or partial distances (PDS) and the average search time for the full-search algorithm, the standard PDS algorithm, and the proposed SCPS algorithm.

Table 1: Average number of computed (partial) distances and relative average search time.

$d=14$ $N=512$ $k=6$	Average number of computed distances	Relative VQ cost
Full search	512	100
Full search - PDS	512	76
SCPS ( $N_1=64$ $k_1=2$ )	64 + 71	28
SCPS ( $N_1=16$ $k_1=3$ )	16 + 91	20

The best results were obtained with the codebook of size  $N_1=16$  and  $k_1=3$  for the first stage. In this case the PDS method did not improve the performance of the first

quantizer, but it added an significant reduction to the cost of the search in the final second probability-sorted subcodebook.

## 5. Conclusions

This work introduced a new fast  $k$ -nearest-neighbors algorithm for vector quantizers called SCPS, and studied its application to speech recognition. The proposed SCPS algorithm has two stages. The first stage uses the  $k_1$  nearest indexes of a small quantizer and a lookup table to select the subset of code vectors to be searched. Then, the second stage performs a PDS search restricted to this subset of the code vectors of interest to obtain the final list of  $k$  nearest code vectors.

The proposed approach can be viewed as an extension of the subcodebook searching (SCS) approach proposed by Tai et al in [10] and the fine-coarse vector quantization proposed by Moayeri [14]. For the first partition, we proposed here the use of two or more indexes of a coarse quantizer to obtain a moderately-fine initial partition but large structured codebooks can be considered as well for this first stage. Additionally, finer partitions and larger lookup tables can be considered to obtain better performance at the expenses of significant storage requirements.

The algorithm was applied to different speech recognition tasks. The results show that the computational cost of the VQ can be significantly reduced without affecting the performance of the speech recognition systems.

## 6. References

- [1] A. Gersho and R. M. Gray, *Vector Quantization and Signal Compression*. Kluwer Academic Publishers, Boston, 1992.
- [2] J. McNames, "A Fast Nearest-Neighbor Algorithm Based on a Principal Axis Search Tree," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 23, no. 9, Sep. 2001.
- [3] D.Y. Cheng, A. Gersho, B. Ramamurthi, and Y. Shoham, "Fast Search Algorithms for Vector Quantization and Pattern Matching," *Proc. IEEE Intl. Conf. Acoustics, Speech, and Signal Processing*, vol. 1, pp. 9.11.1-9.11.4, Mar. 1984.
- [4] J. McNames, "Rotated Partial Distance Search for Faster Vector Quantization Encoding," *IEEE Signal Processing Letters*, vol. 7, no. 9, Sep. 2000.
- [5] L. Torres, J. Huguet, "An improvement on codebook search for vector quantization," *IEEE Trans. Communications*, vol. 42, no. 2, pp. 208–210, Feb. 1994.
- [6] E. Vidal, "An algorithm for finding nearest neighbors in approximately constant average time complexity," *Pattern Recognition Letters*, vol. 4, pp 145-157, 1986.
- [7] L. Guan and M. Kamel, "Equal-average hyperplane partitioning method for vector quantization of image data," *Pattern Recognition Letters*, pp. 693–699, 1992.
- [8] S.W. Ra and J. K. Kim, "A fast mean-distance-ordered partial codebook search algorithm for image vector quantization," *IEEE Trans. Circuits Systems*, vol. 40, pp. 576–579, Sept. 1993.
- [9] C.Y. Chen, C.C. Chang, and R.C.T. Lee, "A Near Pattern-Matching Scheme Based Upon Principal Component Analysis," *Pattern Recognition Letters*, vol. 16, pp. 339-345, Apr. 1995.
- [10] S.C. Tai, C.C. Lai, and Y.C. Lin, "Two Fast Nearest Neighbor Searching Algorithms for Image Vector Quantization," *IEEE Trans. Communications*, vol. 44, no. 12, pp. 1623-1628, Dec. 1996.
- [11] A. Buzo, A. H. Gray Jr., R. M. Gray and J. D. Markel, "Speech coding based upon vector quantization," *IEEE Trans. Acoustics, Speech, and Signal Processing*, vol. 28, pp 562-574, Oct. 1980.
- [12] B.H. Juang, A.H. Gray, "Multiple Stage Vector Quantization for Speech Coding," *Proc. IEEE Intl. Conf. Acoustics, Speech, and Signal Processing*, pp. 597-600, Mar. 1982.
- [13] H.C. Woo, and T. P. Barnwell III, "Fast codeword search for vector quantization using a multi-stage approach," *Proc. IEEE Intl. Conf. Acoustics, Speech, and Signal Processing*, vol. 5, pp. 2629-2632, June 2000.
- [14] N. Moayeri, D. L. Neuhoff, and W. E. Stark, "Fine-Coarse Vector Quantization," *IEEE Trans. Signal Processing*, vol. 39, no. 7, pp. 1503-1515, July 1991.
- [15] SpeechDat. Speech Databases for the creation of voice driven teleservices. EC Telematics. Language Engineering Resources. <http://www.speechdat.org>.