

Fast codebook search algorithm for unconstrained vector quantisation

C.-Q. Chen
S.-N. Koh
I.-Y. Soon

Indexing terms: Fast search, Vector quantisation, Speech coding

Abstract: Vector quantisation (VQ) is a well known data compression technique which maps an ordered set of real numbers into a single integer. However, it is difficult to achieve accurate compression with the use of unconstrained Voronoi VQ when the codebook level and vector dimensionality are very large, due to the extremely high real-time computational complexity involved in full codebook search. To overcome this difficulty, a classified pre-selection method is proposed. Compared to the conventional full search method, the algorithm reduces the computational complexity involved in the code vector selection procedure by 70% ~ 90% with almost no loss in coder performance, at the cost of only a slight increase in the storage requirement.

1 Introduction

In the use of the unconstrained Voronoi vector quantiser, an input vector is evaluated in terms of its similarity to each of a set of code vectors. The code vector that matches the input vector best is used as an approximation to the input, and only its index is transmitted. Thus, computational complexity involved in codebook search is proportional to kN [1, 2], where k is the vector dimensionality and N is the codebook level. Applications of unconstrained Voronoi vector quantisation (VQ) for accurate representation of signal vectors with high dimensionalities have consequently been impeded, although vector quantisation has the potential to achieve coding performance close to the rate-distortion limit with an increasing value of k . To circumvent this obstacle, a fast search algorithm is usually used to simplify the code vector selection procedure. Such an algorithm is formulated to optimise both the coder performance and real-time computational complexity. Various algorithms concerning fast codebook search [1–18] have been suggested previously, such as the partial distance search, peano scanning par-

tial distance search, mean difference and eigen vector methods.

By partially accumulating distortion component by component between an input vector and a code vector, the PDS [3, 4] method rejects the code vector as soon as the hitherto minimum mean square error (MSE) is exceeded. As it is almost certain to obtain a good candidate code vector before the end of the search procedure through the codebook, the PDS method usually reduces the number of multiplications involved in the evaluation process. It thus saves some calculations without degrading the coder performance by taking advantage of the relative simplicity of a logical comparison. The peano scanning partial distance search method [17], mean difference and eigen vector methods [15] are based on the PDS method. By the adoption of a preprocessor operating on the features extracted from the original vectors, each method gains some advantages over the original PDS method. As such methods only use local information such as the specified features, hitherto minimum MSE and the currently accumulated distortion of components, without taking into account global information such as the overall distribution of the code vectors and input vectors, they are very likely to lead to solutions which are locally optimal in terms of real-time complexity minimisation.

The K-d tree [10] is perhaps the most widely used basis for the development of fast search algorithms. The bucket-Voronoi intersection algorithm with the optimised configuration of the K-d tree [5, 9, 11] can achieve a very high efficiency in codebook search. However, this scheme needs to have the explicit geometric description of each Voronoi cell, directly or indirectly, and thus leads to a considerable increase in storage demand. Furthermore, the requirement of the explicit Voronoi region information of each code vector contradicts the definition of a nearest neighbour or Voronoi vector quantiser [1, 2], which is of particular interest.

2 Basic concepts of the classified pre-selection method

Usually, there are some correlations among the different components of a signal vector. In the clustering procedure of codebook generation, each code vector, which is the centroid of each cluster of training vectors, naturally inherits the component correlations from the training set. As a result, the code vectors in a codebook tend to assume some patterns, and thus the code vectors belonging to each pattern tend to match an input

© IEE, 1998

IEE Proceedings online no. 19981691

Paper first received 22nd November 1996 and in revised form 8th September 1997

The authors are with the School of Electrical and Electronic Engineering, Nanyang Technological University, Nanyang Avenue, Singapore 639798

vector in a similar way. Based on the above observations, a classified pre-selection (CP) method has been developed to reduce the real-time computational complexity involved in the use of an unconstrained Voronoi vector quantiser. For a given codebook, the CP method classifies all the code vectors into several clusters. The centroid of each code vector cluster is evaluated in terms of its closeness to an input vector. Those clusters of code vectors whose centroids match the input vector best are chosen as the candidates. Finally, the code vectors belonging to the pre-selected candidate clusters are fully searched or completely evaluated to find the most representative one. Therefore, the proposed CP method may be divided into two successive phases: pre-selecting the candidate code vector clusters for the approximation to each input; and determining the most representative among the code vectors included in the pre-selected candidate clusters.

For example, as shown in Fig. 1, there are nine clusters formed from a given set of two-dimension code vectors. The Euclidean distance between an input X and each code vector cluster centroid Z_i is calculated for $i = 1, 2, \dots, 9$. The three centroids, Z_4 , Z_5 and Z_6 , are then identified as they are closest to X . Finally, the code vectors included in the clusters with the identified centroids Z_4 , Z_5 and Z_6 are fully evaluated to find the most representative for the input vector X .

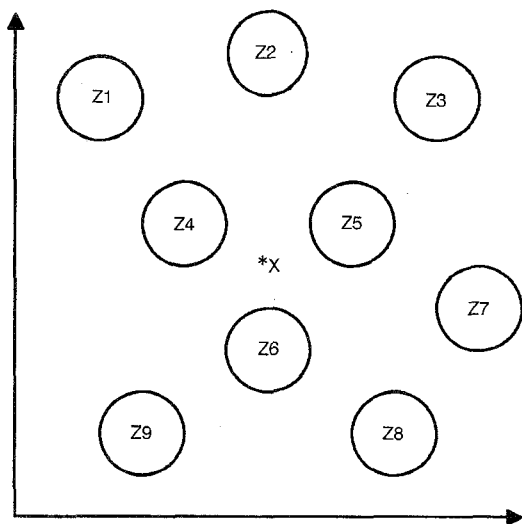


Fig. 1 Typical selection of the candidate clusters with centroids Z_4 , Z_5 and Z_6 from nine code vector clusters for an input vector X

The proposed CP method is developed on the belief that the greater the total number of code vector clusters within a given codebook, the more similarly the code vectors belonging to each cluster match an input vector and the higher is the pre-selection reliability. It is also developed on the basis that the more candidate clusters there are, the better is the resulting coder performance for a given classification of code vectors. However, a greater total number of code vector clusters inevitably corresponds to a greater computational demand associated with the pre-selection phase, and a greater number of candidate code vectors always leads to greater real-time computational complexity involved in the subsequent determination phase. Therefore, there is a trade-off between the real-time computational complexity and the coder performance associated with the use of the proposed CP method. To balance such a trade-off, the total number of code vector clusters and

the number of candidate code vector clusters should be appropriately chosen.

The other bases on which the proposed CP method is developed include the apparent facts that for a given number of candidate code vector clusters, a smaller total number of code vector clusters usually results in a better coder performance; and that the fewer the total code vector clusters and the pre-selected candidate code vector clusters there are, the smaller the number of logical operations involved in finding the closely matching code vector clusters for each input vector.

For a given codebook level N , the number of possible configurations of our CP method is equal to $\sum_{i=1}^N i^2$. Hence, the sum total of the computational amounts involved in exactly evaluating all the possibilities, in terms of the actual resulting coder behaviours, tends to be extremely large when the codebook level and dimensionality are high. However, the computational complexity involved in code vector classifications and evaluations of coder behaviour associated with a specified range of the number of pre-selected candidate code vector clusters is at an acceptable level. Consequently, it is unfeasible to obtain the optimal configuration of the proposed CP method for a high-level and high-dimension codebook using the thorough evaluation approach.

To solve the above problem, we propose that an indicator, the speed index, is used as an alternative to the conventional computational complexity measure. We have been led to this proposal by two main considerations. First, the overall computational complexity is proportional to the average number of code vector components involved in the evaluation process for each input. Secondly, the computational complexity ratio of the conventional full search method to a fast search algorithm describes the speed improvement succinctly. The speed index measure S is therefore defined as

$$S = \frac{Nk}{A} \quad (1)$$

where N , k , and A are the codebook level, the signal vector dimensionality and the average number of code vector components involved in the evaluation process for each vector inside a given input set, respectively. As S attempts to indicate the 'fastness' of a fast search algorithm, it describes more vividly the algorithm behaviour than the number of arithmetic and logical operations.

As described above, only the centroids of all the code vector clusters and the code vectors involved in the pre-selected candidate clusters are evaluated in the use of the proposed CP method. As the average number of code vectors belonging to each cluster is equal to the codebook level divided by the total number of code vector clusters, it is somewhat reasonable to define S_c , a CP version of speed index, as

$$S_c = \frac{N}{(T + CN/T)} = \frac{1}{(T/N + C/T)} \quad (2)$$

where T and C are the numbers of all code vector clusters and candidate clusters, respectively.

It is apparent from the above definitions that the reduced complexity tends to be inversely proportional to the value of S_c and that S_c reaches its maximum value of $0.5 \cdot (N/C)^{1/2}$ if $T = (C \cdot N)^{1/2}$. Therefore, the CP method can be optimised or nearly optimised by only enumerating a range of C values and the neighbourhood of the T value of $(C \cdot N)^{1/2}$ in terms of the resulting

coder performance and computational complexity. Such a mechanism may be explained by the fact that there are more than one, but less than two, degrees of freedom in the configuration optimisation of our CP method. This is because those configurations in which the values of C are close to those of T invariably incur high real-time complexity, and thus should definitely be discarded. It is also apparent that a larger codebook level N will lead to a larger maximum value of S_c , and consequently a larger reduction in real-time computational complexity. Moreover, a smaller value of C will lead to a larger maximum value of S_c , and thus a larger expected reduction in real-time complexity, which is concomitant with a drop in average distortion. A typical function of S_c against T and C for $N = 2048$ is shown in Fig. 2. As shown, the values of S_c are large only within a small region spanned by T and C .

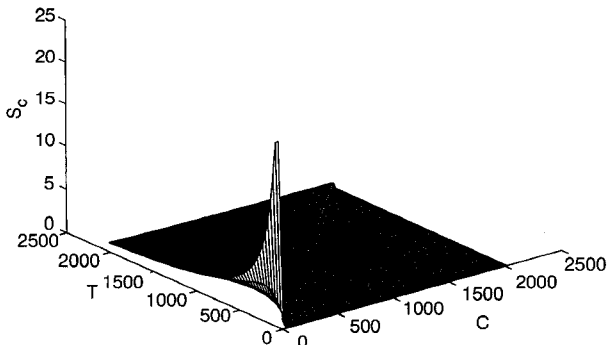


Fig. 2 CP version of speed index S_c against T , the total number of code vector clusters and C , the number of candidate clusters, at $N = 2048$

In our opinion, there are three approaches to real-time complexity reduction for codebook search in the use of unconstrained Voronoi vector quantiser along which research efforts may be meaningfully directed. The first approach is to exploit the code vector component correlations inherited from the training set used for codebook generation. The second approach is to determine some exploitable features of the original signal, and then reject those code vectors that are far from the input vector in the extracted feature space. The third approach is to simplify the real-time procedure of feature extraction used in the second approach. As the centroid of each cluster may be regarded as the feature vector extracted from each subset of code vectors, and each input may also be considered as a feature vector itself, the proposed CP method combines all three approaches. It is notable that PDS-based methods are very useful in determining the most representative code vector among a set of candidates, and all three approaches are usually used to discard or keep some code vectors in order to select the appropriate candidate code vectors for each input vector.

3 Classified pre-selection method

With a given test set $V\{v_i; i = 1, 2, \dots, U\}$ and an N -level codebook $Y\{y_i; i = 1, 2, \dots, N\}$ generated from a training set $X\{x_i; i = 1, 2, \dots, M; M \gg N\}$ of k -dimension vectors using the LBG algorithm [19], the design and encoding procedures of the proposed CP method can be stated as below.

3.1 Design procedure

Step 1: set C , the number of pre-selected candidate clusters, to be 1.

Step 2: calculate T , the total number of code vector clusters, according to $T = (C \cdot N)^{\frac{1}{2}}$.

Step 3: classify all the code vectors y_i ($i = 1, 2, \dots, N$) in the given codebook Y into T clusters, R_1, R_2, \dots, R_T ($\bigcup_{j=1}^T R_j = Y$), using a modified LBG algorithm (which is slightly different from the original version in that only the cluster of code vectors with the maximum distortion, instead of all the clusters, is split at the beginning of each stage). Set the input index i to be 1.

Step 4: calculate the Euclidean distance $d(x_i, z_j)$ between the current training vector x_i and z_j , the centroid of code vector cluster R_j formed in Step 3, for $j = 1, 2, \dots, T$. Select the C code vector clusters R_l^* ($l = 1, 2, \dots, C$) whose centroids z_l^* ($l = 1, 2, \dots, C$) match x_i most closely, as the candidate code vector clusters for the approximation to x_i :

$$d(x_i, z_l^*) \leq d(x_i, z_j) \text{ for all } z_j \notin \{z_l^* : l = 1, 2, \dots, C\}$$

Step 5: pick up and then fully evaluate the code vectors grouped under the C candidate code vector clusters selected in Step 4, to find the most representative one for x_i :

$$\hat{x}_i = \min_{y_j}^{-1} d(x_i, y_j) \text{ for } y_j \in \bigcup_{l=1}^C R_l^*$$

Increase i by 1. Go to Step 4 if i is not greater than M . Otherwise, calculate the average Euclidean distortion associated with each training vector x_i and its resulting reproduction \hat{x}_i for $i = 1, 2, \dots, M$.

Step 6: calculate the difference between the average Euclidean distortion obtained at the end of Step 5 and that associated with the conventional full search method for the given training set. Go to Step 8 if such a difference is less than the product of a pre-determined value δ and the average Euclidean distortion associated with the conventional full search method. Otherwise, go to Step 7.

Step 7: decrease T by 1. Go to Step 3 if T is greater than T_m , where T_m is set to be $0.5 \cdot (C \cdot N)^{\frac{1}{2}}$ on the basis that S_c reaches its maximum value if $T = (C \cdot N)^{\frac{1}{2}}$ and $T \propto S_c$ if $T < (C \cdot N)^{\frac{1}{2}}$.

Step 8: store the current values of T , C , and code vector classification $Y\{R_j; j = 1, 2, \dots, T\}$, if the resulting computational complexity is the lowest so far. Increase C by 1. Go to step 2 if C is not greater than a positive integer C_m , which may be set to be proportional to $\log_2 N$. Otherwise, stop.

3.2 Encoding/test procedure

Step 1: calculate the Euclidean distance $d(v_i, z_j)$ between each input vector v_i inside the given test set $V\{v_i; i = 1, 2, \dots, U\}$ and z_j , the centroid of code vector cluster R_j , for $j = 1, 2, \dots, T$. Select the C code vector clusters whose centroids z_l^* ($l = 1, 2, \dots, C$) match v_i most closely, as the candidate code vector clusters for the approximation to v_i :

$$d(v_i, z_l^*) \leq d(v_i, z_j) \text{ for all } z_j \notin \{z_l^* : l = 1, 2, \dots, C\}$$

where the values of T , C and code vector clusters R_j ($j = 1, 2, \dots, T$) were obtained in the above design procedure.

Step 2: pick up and then fully evaluate the code vectors grouped under the C candidate code vector clusters R_l^* ($l = 1, 2, \dots, C$) selected in Step 1, to find the most

representative one for v_i :

$$\hat{v}_i = \min_{y_j}^{-1} d(v_i, y_j) \text{ for } y_j \in \bigcup_{l=1}^C R_l^*$$

It should be noted that the training set is used to generate the codebook and then design the configuration of the proposed CP method, whereas the test set is used to assess the behaviour of the CP method. Obviously, the nearly optimised CP method is configured by the values of T , C and code vector classification $Y\{R_j; j = 1, 2, \dots, T\}$ thus obtained. In an attempt to obtain the optimised version of our CP method, a neighbourhood in which the value of T is greater than $(C \cdot N)^{1/2}$ must also be evaluated in terms of the actual resulting coder performance and computation requirement. Notably, the membership-weighted average should be used to substitute the unweighted average N/T in calculating the speed index of the proposed CP method, in order to describe the coder behaviour inside a training set or a test set more accurately. The proposed index S_c is a simplicity-oriented version and is used only as a design parameter aid.

It is important to emphasise that the proposed CP method will achieve exactly the same performance as the conventional full search method as long as the value of C is high enough; the corresponding configuration can be obtained by simply setting the value of the performance difference allowance δ to be zero in the above design procedure. However, we focus on the conditional minimisation of computational complexity, i.e. reducing the real-time computational complexity as much as possible, while ensuring that the average distortion level is satisfactorily maintained.

4 Normalised computational complexity

To effectively design the configuration and roughly quantify the advantage of the proposed CP method, a normalised computational complexity measure W is presented as the ratio of the average computational amount of multiplications and additions involved in the encoding of each of the L vectors, inside a given input set, using our fast search algorithm to that using the conventional full search method. The definition and derivation of W are presented below.

Let the number of code vectors belonging to the i th code vector cluster be c_i , and the number of vectors inside the given input set that incur the full evaluation of the i th code vector cluster be m_i ; $\sum_{i=1}^T m_i = C \cdot L$. The total number of code vectors involved in the full evaluation for the encoding of each input vector is then given by $T + (1/L)\sum_{i=1}^T m_i \cdot c_i$. On the other hand, each input vector incurs the complete evaluation of N code vectors in the use of the conventional full search method. Therefore

$$\begin{aligned} W &= \left(T + \frac{1}{L} \sum_{i=1}^T m_i c_i \right) k / (Nk) \\ &= \frac{T}{N} + \frac{1}{LN} \sum_{i=1}^T m_i c_i \end{aligned} \quad (3)$$

The normalised computational complexity measure W is obviously more informative and sensitive than the speed index S_c for indicating the behaviour of the proposed CP method. The actual average amount of computation involved in using the proposed CP method for encoding an input vector is equal to W multiplied by

the computational amount associated with the conventional full search method, plus that of the additional logical operations whose number is less than $T \cdot (C - 1)$.

5 Application of the CP method for LPC codebook search

Linear predictive coding (LPC) parameters are widely used in speech coders for conveying the spectral envelope information. For low bit rate quantisation, it is important to encode these parameters to achieve as low a value as possible for the product, (number of bits) \times distortion, with acceptable computational complexity and storage requirements. Hence, a fast search algorithm usually plays an important role in encoding the LPC information.

In our application of the proposed CP method for LPC codebook search, an LPC vector is made up of ten line spectral frequencies (LSFs) [20], which are transformed from the LPC coefficients obtained through the tenth-order LPC analysis [21, 22]. The training and test data sets consist of a selection of speech signals from the TIMIT database [23]. The sampling frequency is 8kHz, and the analysis frame rate is 50 frames per second with 160 samples per frame. For different codebook levels and different training sets, code vectors are generated using the LBG algorithm [19]. With each of the specified values of difference allowance percentage of coder performance δ , the nearly optimised version of the proposed CP method is designed; its behaviour is then evaluated by checking the resulting computational complexity and coder performance with those associated with the conventional full search method for each of the generated codebooks. All simulation programmes were written in C, and the distortion of the encoded LPC parameters inside and outside the training set was measured in terms of the average Euclidean distortion (AED) and the average spectral distortion (ASD) [24], which are given as

$$AED = \frac{1}{U} \sum_{i=1}^U \sum_{j=1}^k (x_{ij} - y_j)^2 \quad (4)$$

$$ASD = \frac{1}{U} \sum_{i=1}^U \left[\frac{1}{F_s} \int_0^{F_s} [10 \log_{10} P_i(f) - 10 \log_{10} \hat{P}_i(f)]^2 df \right]^{1/2} \quad (5)$$

where x_{ij} and y_j are the j th components of the i th input vector and corresponding code vector; $P_i(f)$ and $\hat{P}_i(f)$ are the LPC power spectra of the i th frame associated with the original and reconstructed LPC information, respectively; F_s is the sampling frequency; and U is the size of test set.

Tables 1–5 show that the real-time computational complexity involved in codebook search using the nearly optimised CP method is much less than that using the conventional full search method, and the difference in performance between the CP method and the conventional full search method is less than each of the predetermined allowances. As expected, the computational complexity reduction is increased when a larger value of difference allowance percentage of coder performance is specified, and/or when a higher codebook level is used.

Table 1: Behaviour comparison of CP method and full search method for $N = 256$

	CP method	CP method	CP method	Full search method
Difference allowance	$\delta = 0.015$	$\delta = 0.01$	$\delta = 0.005$	-----
Total number of code vector clusters T	27	21	31	512
Candidate cluster number C	3	3	4	512
W for training set	0.222441	0.232905	0.263852	1
W for test set	0.222381	0.233565	0.26379	1
AED for training set	0.000789	0.000787	0.000783	0.00078
ASD for training set (dB)	2.826803	2.823632	2.819345	2.814871
AED for test set	0.000919	0.000916	0.000912	0.000908
ASD for test set (dB)	3.015928	3.010908	3.007914	3.000718

W = normalised computational complexity, size of training set = 4336, size of test set = 4336

Table 2: Behaviour comparison of CP method and full search method for $N = 512$

	CP method	CP method	CP method	Full search method
Difference allowance	$\delta = 0.015$	$\delta = 0.01$	$\delta = 0.005$	-----
Total number of code vector clusters T	34	45	31	512
Candidate cluster number C	3	4	4	512
W for training set	0.16373	0.189024	0.201969	1
W for test set	0.163783	0.189062	0.202089	1
AED for training set	0.000667	0.000663	0.00066	0.000658
ASD for training set (dB)	2.614236	2.605645	2.602936	2.598277
AED for test set	0.000793	0.000789	0.000787	0.000783
ASD for test set (dB)	2.83215	2.824873	2.820744	2.817255

W = normalised computational complexity, size of training set = 9168, size of test set = 9168

Table 3: Behaviour comparison of CP method and full search method for $N = 1024$

	CP method	CP method	CP method	Full search method
Difference allowance	$\delta = 0.015$	$\delta = 0.01$	$\delta = 0.005$	-----
Total number of code vector clusters T	64	63	66	1024
Candidate cluster number C	4	4	4	1024
W for training set	0.131996	0.132505	0.147207	1
W for test set	0.132071	0.132629	0.147272	1
AED for training set	0.00054	0.000539	0.000537	0.000534
ASD for training set (dB)	2.354371	2.3539	2.349719	2.345638
AED for test set	0.000657	0.000657	0.000654	0.000651
ASD for test set (dB)	2.576609	2.576265	2.571056	2.566975

W = normalised computational complexity, size of training set = 19098, size of test set = 19098

Table 4: Behaviour comparison of CP method and full search method for $N = 2048$

	CP method	CP method	CP method	Full search method
Difference allowance	$\delta = 0.015$	$\delta = 0.01$	$\delta = 0.005$	-----
Total number of code vector clusters T	79	66	74	2048
Candidate cluster number C	4	4	5	2048
W for training set	0.095957	0.103615	0.113749	1
W for test set	0.09555	0.102932	0.113166	1
AED for training set	0.000497	0.000496	0.000494	0.000491
ASD for training set (dB)	2.211323	2.209756	2.205761	2.201644
AED for test set	0.000579	0.000579	0.000576	0.000574
ASD for test set (dB)	2.383903	2.38253	2.378873	2.374319

W = normalised computational complexity, size of training set = 70660, size of test set = 17665

It is notable that using the proposed CP method will incur a slight increase in storage requirements compared to the conventional full search method. The increase in memory requirement is equal to $k \cdot T$

real numbers for storing the centroid of every code vector cluster, plus T integers for storing the code vector number associated with each code vector cluster.

Table 5: Behaviour comparison of CP method and full search method for $N = 4096$

	CP method	CP method	CP method	Full search method
Difference allowance	$\delta = 0.015$	$\delta = 0.01$	$\delta = 0.005$	-----
Total number of code vector clusters T	125	143	156	4096
Candidate cluster number C	4	5	6	4096
W for training set	0.067647	0.076089	0.082424	1
W for test set	0.067204	0.0756	0.082013	1
AED for training set	0.0004	0.000398	0.000397	0.000395
ASD for training set (dB)	2.00425	1.999728	1.997509	1.993936
AED for test set	0.000506	0.000504	0.000503	0.000501
ASD for test set (dB)	2.249362	2.244937	2.243228	2.239223

W = normalised computational complexity, size of training set = 70660, size of test set = 17665

6 Conclusions

The performance and computational complexity of the proposed CP method have been compared with those of the conventional full search method for a given set of input vectors and a codebook generated from a training set. The performance measures used are the average Euclidean distortion and spectral distortion. The CP method is found to be much simpler than the conventional full search method, while achieving almost the same coder performance. More improvements are expected if methods such as partial distance search and triangle-inequality elimination [25, 26] are employed instead of the complete distance search process involved in both the pre-selection and determination phases, and/or if the procedure of pre-selection followed by determination is applied to the candidate pre-selection phase itself in the proposed CP method; a multi-layer CP method may outperform the two-layer version described in this paper.

The other advantages of the novel CP method include

- (i) the configuration design procedure is simple compared with the K-d tree-based schemes
- (ii) the worst case real-time complexity is relatively low due to the constant number of candidate code vector clusters selected for encoding each input vector
- (iii) the explicit geometric description of Voronoi cell associated with each code vector is not required.

The improved encoding speed of the proposed CP method may lead to the practical implementation of unconstrained nearest neighbour vector quantisation with a high-dimensionality and a very high codebook level, as the reduction in real-time complexity is proportional to the codebook level. In addition, the new concept concerning code vector classification, speed index and normalised computational complexity measures may establish a bearing point for future investigations.

7 References

- 1 GERSHO, A., and GRAY, R.M.: 'Vector quantization and data compression' (Kluwer, Boston, 1992)
- 2 GRAY, R.M.: 'Vector quantization', *IEEE ASSP Mag.*, 1984, 1, pp. 4-29
- 3 BEI, C.D., and GRAY, R.M.: 'An improvement of minimum distortion encoding algorithm for vector quantization', *IEEE Trans.*, 1985, C-33, (10), pp. 1132-1133
- 4 CHENG, D.Y., GERSHO, A., RAMAMURTHI, B., and SHOHAM, Y.: 'Fast search algorithms for vector quantization and pattern matching'. IEEE Proc. ICASSP, 1984, pp. 9.11.1-9.11.4

- 5 CHENG, D.Y., and GERSHO, A.: 'A fast codebook search algorithm for nearest neighbour pattern matching'. IEEE Proc. ICASSP, 1986, pp. 265-268
- 6 SOLEYMANI, M.R., and MORGERA, S.D.: 'A fast MSE encoding for vector quantization', *IEEE Trans.*, 1989, C-37, pp. 656-659
- 7 HUANG, C.M., BI, Q., STILES, G.S., and HARRIS, R.W.: 'Fast full search equivalent encoding algorithms for image compression using vector quantization', *IEEE Trans. Image Process.*, 1992, 1, (3), pp. 413-416
- 8 PALIWAL, K.K., and RAMASUBRAMANIAN, V.: 'Effect of ordering the codebook on the efficiency of the partial distance search algorithm for vector quantization', *IEEE Trans. Inf. Theory*, 1989, 15, pp. 658-664
- 9 LOWRY, A., HOSSAIN, S., and MILLAR, W.: 'Binary search trees for vector quantization'. Proc. ICASSP, 1987, pp. 2205-2208
- 10 BENTLEY, J.L.: 'Multidimensional binary search trees used for associative searching', *Commun. ACM*, 1975, 18, (9), pp. 209-226
- 11 RAMASUBRAMANIAN, V., and PALUWAL, K.K.: 'Fast k-dimensional tree algorithms for nearest neighbour search with application to vector quantization encoding', *IEEE Trans. Signal Process.*, 1992, 40, (3), pp. 518-531
- 12 RAMASUBRAMANIAN, V., and PALUWAL, K.K.: 'An efficient approximation elimination algorithm for fast nearest neighbour search based on a spherical distance co-ordinate formulation', *Pattern Recog. Lett.*, 1992, 13, pp. 471-480
- 13 SOLEYMANI, M.R., and MORGERA, S.D.: 'An efficient nearest neighbour search method', *IEEE Trans.*, 1987, C-35, (6), pp. 677-679
- 14 ORCHARD, M.T.: 'A fast nearest neighbour search algorithm'. IEEE Proc. ICASSP, 1991, pp. 2297-2300
- 15 LEE, C.-H., and CHEN, L.-H.: 'High-speed closest codeword search algorithms for vector quantization', *Signal Process.*, 1995, 43, pp. 323-331
- 16 CHANG, L., and BAYOUMI, M.M.: 'An economical binary tree structure for vector quantization'. Proc. ICASSP 91, Ontario, Canada, 14-17 April 1991, Vol. 1, pp. 669-672
- 17 QUWEIDER, M.K., and SALARI, E.: 'Peano scanning partial distance search for vector quantization', *IEEE Signal Process. Lett.*, 1995, 2, (9), pp. 169-171
- 18 VIDAL, E.: 'An algorithm for finding nearest neighbours in (approximately) constant average time complexity', *Pattern Recog. Lett.*, 1986, 4, pp. 145-157
- 19 LINDE, Y., BUZO, A., and GRAY, R.M.: 'An algorithm for vector quantizer design', *IEEE Trans.*, 1980, C-28, (1), pp. 84-95
- 20 ITAKURA, F.: 'Line spectrum representation of linear predictive coefficients of speech signals', *J. Acoust. Soc. Amer.*, April 1975, 57, pp. s35
- 21 CUMANI, A.: 'On a covariance-lattice algorithm for linear prediction'. Proc. ICASSP, 1982, Vol. II, pp. 615-654
- 22 Vector Sum Excited Linear Prediction 11200 Bit Per Second Voice Coding Algorithm, Chicago Corporate Research and Development Center, Motorola Inc., USA, 1990
- 23 FISHER, W.M., DODDING, G.R., and GOUDIE-MARSHALL, K.M.: 'The DARPA speech recognition research databases: specifications and status'. Proc. DARPA Speech Recognition Workshop, February 1986, pp. 54-60 (Report Saic-86/1546)
- 24 GRAY, A.H., and MARKEL, J.D.: 'Distance measures for speech processing', *IEEE Trans.*, 1976, ASSP-24, pp. 380-391
- 25 SOLEYMANI, SOLEYMANI, M.R., and MORGERA, S.D.: 'An efficient nearest neighbour search method', *IEEE Trans.*, 1987, C-35, (6), pp. 677-679
- 26 VIDAL, E.: 'An algorithm for finding nearest neighbours in (approximately) constant average time complexity', *Pattern Recog. Lett.*, 1986, 4, pp. 145-157