Scientia Iranica, Vol. 6, No. 1, pp 25–31
© Sharif University of Technology, January 1999

SCIENTIA
IRANICA

Research Note

Efficient Spectral Coding of Speech Using Generalized Sorted Codebook Vector Quantization Applied to LSF Parameters

H.R. Sadegh-Mohammadi[1]

Sorted Codebook Vector Quantization (SCVQ) is shown to be a very efficient vector quantization method. This paper presents different approaches for generalizing the SCVQ method, including related algorithms for the training and optimization of the associated codebooks. Then, applications of the generalized SCVQ in spectral coding of speech using the quantization of Line Spectral Frequencies (LSFs), which are the most popular parameters to represent the linear prediction model for spectrum quantization in speech coders, are described. Different experiments were conducted to evaluate the performance of the new method. Lower quantization distortion of the new method is verified through computer simulations.

INTRODUCTION

Finding high quality speech coding methods at low bit rates is a major goal for the speech processing community. One requirement for this achievement is the reproduction of the spectral envelope at the decoder side with very low distortion. The research trend in the spectral coding area is focused on quantization methods that provide the best compromise between low quantization rate, low encoding degradation and high efficiency in the implementation stage.

Spectral coding of speech is one of the most important stages of low rate coders, which use the linear prediction models, such as Code-Excited Linear Prediction (CELP) [1]. Among the various representations of the linear prediction coefficients, Line Spectrum Frequencies (LSFs) [2] are often used for short-term spectrum quantization because of their desirable properties, which are exploited during spectral coding. Various quantization techniques have been developed for LSFs during the last decade, including many Scalar Quantization (SQ) and Vector Quantization (VQ) methods.

Although vector quantizers can achieve lower distortion than scalar quantizers at the same bit rates (or alternatively, realize the same quality at lower bit rates), their application has been restricted by their considerable computational and storage costs. Cost reduction has its price; namely an increase in quantization distortion. Fortunately this quality degradation is minor and now there are VQ methods requiring only medium amounts of memory and moderate computation.

Sorted Codebook Vector Quantization (SCVQ) proposed in [3] is verified to be an efficient method for the quantization of the LSFs. That reference also presents a particular algorithm which optimizes the associated codebooks. Since then, two algorithms have been suggested for selection of a key parameter (called sorting parameter) in the SCVQ method [4,5]. While the performance and implementation costs of SCVQ have proved to be notably impressive, it is still possible to further reduce its quantization distortion by the generalization of this method for spectral coding or other similar applications.

In this article, a generalization of SCVQ using a combination of several approaches is suggested and its application to spectral coding of speech is described.

This paper is organized as follows. In the next section, a brief review of the principles of SCVQ is presented. Then, various approaches are proposed for the generalization of generic SCVQ and generalized training and optimization procedures are referred for design of the related codebook. Moreover, the selection of a set of key parameters in this method (i.e., sorting parameters) is addressed. Simulation experiments are also provided and the results of quantization with different methods are evaluated in terms of different

1. *Electrical Engineering Research Center, Jahad Daneshgahi, Iran University of Science and Technoloy, Tehran, I.R. Iran.*

objective measures. Finally, the results of comparison between the ordinary SCVQ and its generalized counterparts are discussed, followed by the conclusions.

REVIEW OF SCVQ METHOD

Sorted Codebook Vector Quantization (SCVQ) is a low cost VQ method that is outlined here briefly. Given an L-dimensional target vector $\mathbf{F}_t = [f_{1t}, f_{2t}, ..., f_{Lt}]^T$ and a codebook C of size N, a sorting parameter is defined $S_t = g(f_{1t}, f_{2t}, ..., f_{Lt})$, which is a scalar by definition, where $g(.)$ is a suitable function called sorting function, chosen in such a way that neighboring target vectors give the neighboring values of S_t. Of course, in most cases this mapping cannot reflect the neigborhood completely. Then, the indices of the codebook are sorted in ascending order of the sorting parameter for each codevector, according to the vector $\mathbf{S}= [S_1, S_2, ..., S_N]^T$ with $S_1 \le S_2 \le ... \le S_N$.

To code the target vector, in the first step, the quantity S_{in}, which is calculated using the operation of $g(.)$ function on the elements of input vector, is scalar quantized by $\mathbf{S}$. Suppose S_i is the result of scalar quantization, with $1 \le i \le N$. This codevector S_i is called the central codevector and its index i is called the central index. Then, the target vector is vector quantized using an extensive search in the neighborhood of the central index. For instance, only the codevectors with indices within the range of $i - k + 1$ to $i + k$ may be searched, where k is an offset value. Generally, the computational complexity of this method grows linearly with k, which normally is set to be much less than N.

GENERALIZED SORTED CODEBOOK VECTOR QUANTIZATION

To extend the application of SCVQ method and to improve its performance, a Generalized Sorted Codebook Vector Quantization (GSCVQ) method is proposed. This method is a result of the generic SCVQ generalization. While it is possible to define different types of generalization individually, it is preferred here to propose several generalization schemes and combine them into a unified framework. These schemes are introduced in turn.

Generalization Using a Shift Variable

In this generalization approach, a shift variable o is introduced in the search procedures, i.e., in selection of the codevectors located in the neighborhood of the central codevector associated with the central index. In the generic SCVQ method, the brute-force search is performed over the $2k$ codevectors in the vicinity of the central index, i.e., only the codevectors with indices within the range of $i-k+1$ to $i+k$, are being searched. Here, it is suggested that the search is carried on the codevectors with indices within the range of $i+o_i-k+1$ to $i + o_i + k$, where each central index has its own associated shift variable (o_i), which may be different from the others.

It should be noted that this change increases the storage cost slightly. However, there is no need to modify the algorithm when the central index is either less than k or greater than $N - k$, as in the SCVQ method with the equal search strategy [3]. An example of incorporating this generalization method into the SCVQ schematic diagram for the quantization of first three LSFs is depicted in Figure 1. In this example, the shift variable associated with the 69th central index is equal to 3 and the offset value is set equal to 8. The Q_S block represents the scalar quantization stage.

It is noteworthy that the shift table is a look-up table which contains the shift variables associated with all central indices. A simple heuristic algorithm for training this table is as follows. After the training and optimization of the SCVQ codebook with a proper method, such as the correlation algorithm (CA) [3] or the direct selection algorithm (DSA) [5], all training vectors whose central indices are equal to 1 ($i = 1$) are selected.

For this collection, the shift variable is changed from a minimum to a maximum integer threshold and for each particular value the whole quantization process is performed. Total quantization distortion is

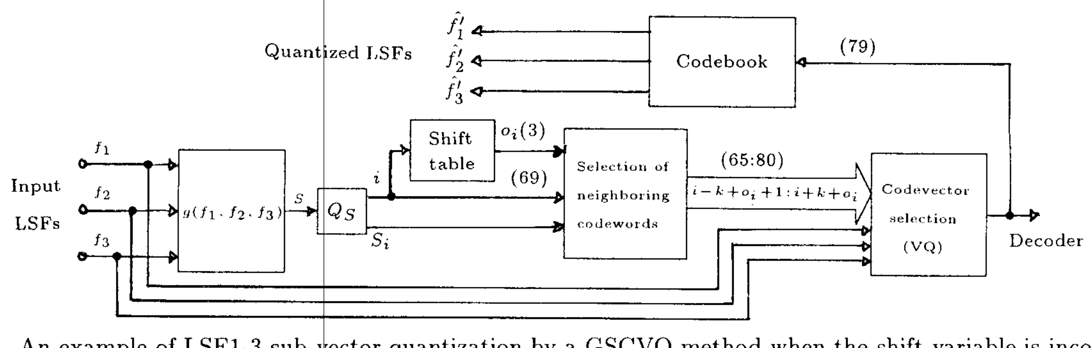


Figure 1. An example of LSF1-3 sub-vector quantization by a GSCVQ method when the shift variable is incorporated into the block diagram.

calculated for all training vectors in the collection and any particular value that minimizes the total distortion is selected as the final value for the shift variable associated with the first codevector. This procedure is continued for the other codevectors in the codebook until the shift variable for the last codevector is chosen. The selected shift variables are stored in a look-up table like the trained shift table.

In general, for some central codewords the described algorithm suggests several shift values as the best choices. This happens especially with small training data sets. In fact, the algorithm proposes a range of integer values. According to previous experiences, in these cases, it is better to consider the integer shift value that is located in the middle of the proposed range as the final choice. It is noteworthy that any suitable distortion measure can be chosen for the calculation of the quantization distortion. A proper range for searching the best shift variable associated with the ith central index is:

$$o_{i_{\min}} \leq o_i \leq o_{i_{\max}} \ ,$$

$$o_{i_{\min}} = \max(o_i - k, 1) \ ,$$

$$o_{i_{\max}} = \min(o_i + k - 1, N). \qquad (1)$$

Generalization Using Neighborhood Dependencies

The second generalization approach is about changing the neighborhood relationship between various codevectors in a small region from one-dimensional space in the index domain to a multi-dimensional dependencies in that domain. In other words, it is possible to replace the neighborhood dependencies between codevectors from a ladder type sorting table to a multi-dimensional hyper-table, such as a two-dimensional rectangular table.

To clarify this statement, a simple diagram is presented which shows the difference between generic and generalized SCVQ using this approach based on the previous example (i.e., $i = 69$ and $k = 8$). Here, the first approach for generalization of SCVQ is not used, so the shift variable is not considered for simplicity. Figure 2 demonstrates an example of neighboring codevectors (the unshaded areas) for generic and generalized SCVQ. In both cases, the final search is performed on $2k$ codevectors in the vicinity of the central codeword.

Different schemes can be employed to choose the codevectors which would be considered as the neighboring codevectors in the quantization process. For example, one may choose the $2k - 1$ codewords as the neighboring codevectors of a particular central codeword whose distances are less than the others. Although this scheme normally provides a very good quantization result, its storage cost increases considerably. However, if this general approach is used properly (e.g., as will be explained later), the additional memory requirement will be insignificant. Moreover, this method can be combined with the first generalization approach in an appropriate manner to reduce the implementation costs. For instance, one may look for the best shift variables after organizing the neighboring dependency.

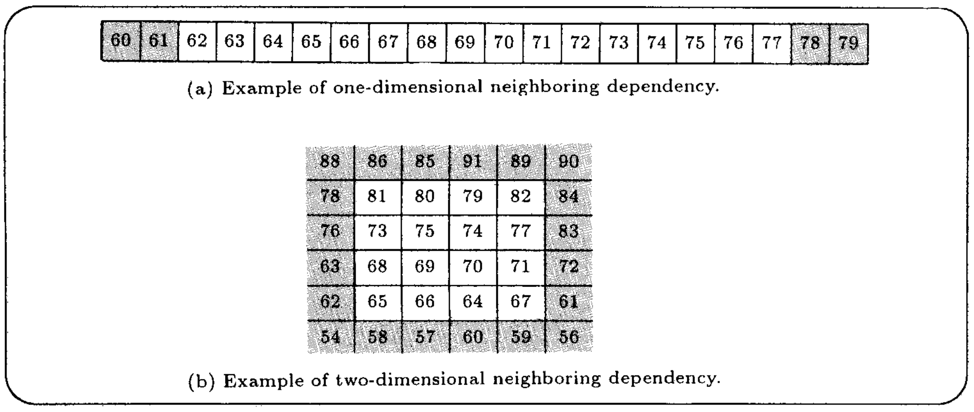
Generalization Using Sorting Function Set

The third approach, which is a major generalization scheme, is about expanding the sorting function in the SCVQ from a single equation to a set of equations. This approach will be discussed extensively later. Here, with no loss of generality, a generalization from one-dimensional to two-dimensional SCVQ is discussed. Of course, the same concepts can be used for generalization to higher dimensions.

Suppose the sorting function set is defined as:

$$\begin{bmatrix} S_t = g_1(f_{1t}, f_{2t}, ..., f_{Lt}) \\ T_t = g_2(f_{1t}, f_{2t}, ..., f_{Lt}) \end{bmatrix}, \qquad (2)$$

where g_1 and g_2 are suitable functions similar to that



60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79

(a) Example of one-dimensional neighboring dependency.

88	86	85	91	89	90
78	81	80	79	82	84
76	73	75	74	77	83
63	68	69	70	71	72
62	65	66	64	67	61
54	58	57	60	59	56

(b) Example of two-dimensional neighboring dependency.

Figure 2. Comparison of one-dimensional and two-dimensional neighboring dependency.

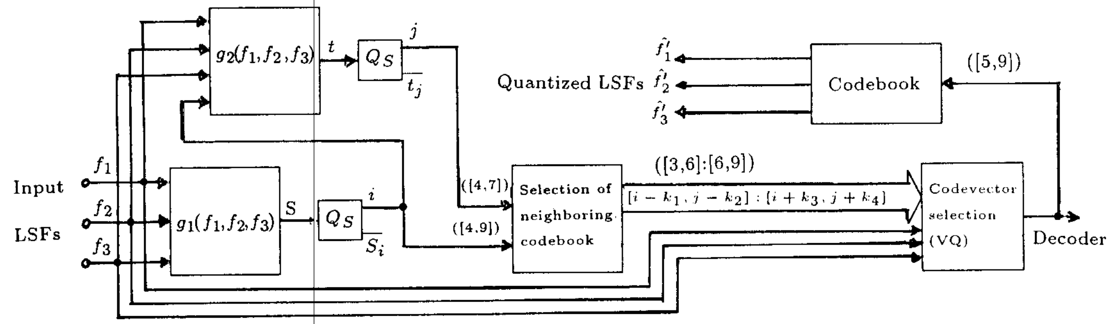


Figure 3. LSF1-3 subvector quantization by a GSCVQ method when a two-dimensional sorting function set is incorporated into the block diagram (the same example as in Figure 1).

of the ordinary SCVQ. Again, S_t and T_t will be known as sorting parameters. The indices of the codevectors in the codebook are sorted in ascending order of the first sorting parameter associated to each codevector, i.e., according to the vector $\mathbf{S} = [S_1, S_2, ..., S_N]^T$ with $S_1 \leq S_2 \leq ... \leq S_N$. These indices are, then, divided consequently into a number of bins, say M bins. The number of codevectors classified to the mth bin is denoted by n_m, which might be either equal to or different from the others. In the next stage, the indices of codevectors associated with each bin are sorted in ascending order of the second sorting parameter for each codevector. This procedure is performed for each bin separately. For instance, this process for a two-dimensional SCVQ with a codebook of 256 codevectors would result in a 16 by 16 table, if both M and n_m are set equal to 16. This table contains the indices of the codevectors and the content of each cell represents the associated central indices. Hereafter, this example will be used for explanations and experiments.

In the quantization process, the sorting parameters for each target vector are calculated. Then, the first computed sorting parameter is compared with the first sorting parameters associated with the 9th row of the table (assuming that the columns of the table are sorted according to the first sorting parameter). Later, the column of the table, whose first sorting parameter results in the minimum distance for the calculated parameter of the target vector is selected. In the next stage, the second sorting parameter computed for the target vector is compared with the second sorting parameters of the codevectors whose indices are located in the selected column of the table. The index of the codevector that provides the best match will be known as the central index (similar to the ordinary SCVQ) and the final classification is performed by a brute-force search among the codevectors whose indices are in the vicinity of the central index. Figure 3 shows a block diagram for GSCVQ with a two-dimensional function set applied to the previous example about LSF1-3 subvector quantization.

Both algorithms (i.e., correlation and direct selection algorithms) that have been suggested for selection of the best sorting function among different functions in the ordinary SCVQ, can be generalized for choosing the best sorting function set among different function sets. However, here just a generalization of the DSA algorithm is explained. In the generalized DSA, a training data set is chosen for performing the algorithm and an unstructured codebook is trained using any proper method (e.g., generalized Lloyd algorithm [6]). For each function set candidate, the codebook is sorted according to what has been mentioned for the GSCVQ previously. Later, each target vector is replaced with its associated central codevector and the total distortion resulting from these replacements is calculated. This procedure is performed for all function sets. Finally, any function set candidate that results in the minimum distortion is chosen as the best.

In a similar generalization, the codebook optimization method presented in [3] can be used for developing a codebook optimization algorithm in GSCVQ.

SIMULATION EXPERIMENTS

Low rate speech coding area is currently dominated by the linear prediction model, e.g., code-excited linear prediction (CELP). The quality of the reproduced speech in these coders depends, to a large extent, on the performance of the spectral coding stage.

In this study, the focus is on a tenth order LP model. Therefore, ten LSFs must be quantized for each frame (30 ms) of the speech signal. Split VQ is considered a good basis for the development of a low cost method for spectrum quantization. The entire LSF vector in each frame of speech (30 ms) is divided into three sub-vectors of dimensions 3, 3 and 4. The LP parameters are extracted in a similar way to the FS1016 CELP standard [1]. This standard is also used for the simulation with quantized LSF values. The proposed SCVQ method has been used for quantization of each sub-vector as well as other vector quantization methods.

The codebooks used for this simulation were trained on the same training data set and similar conditions were applied to the test speech signals.

Various codebooks have been trained on a database of 10,240 LSF vectors extracted from almost 5 minutes of speech signal. Two thirds of the training set was taken from the TIMIT database and the rest from other speech sample sources. The same number of sentences were taken from male and female speakers. Each speaker pronounced just one sentence. The LSF extraction method is as follows: First, the 8 kHz sampled speech signals are divided into 30 ms non-overlapping frames (240 samples). The speech samples in each frame are passed through a 30 ms Hamming window centred on the middle of the frame. Then, linear prediction coefficients are calculated by the autocorrelation method without using preemphasis. For each frame, the LP coefficients are converted to ten LSF parameters. The LSF vectors are divided into three subvectors of lengths 3, 3 and 4, while separate codebooks are generated for each subvector.

Several codebooks have been trained for Unstructured Vector Quantization (UVQ); Tree-Searched VQ (TSVQ), Sorted Codebook VQ (SCVQ) and Generalized VQ (GSCVQ). All these VQs use codebooks of size 256. Hence, the overall number of bits needed for the spectral coding stage is 24 bits/frame. In the SCVQ method, the chosen sorting function was simply the sum of the elements in each subvector and the offset value for the final codebook search was $k = 8$.

In the first experiment, a SCVQ which uses the sum of the elements in each subvector as the sorting parameter is employed. The training and optimization of the codebooks of this SCVQ are explained in [3]. Then, the first generalization approach (using a shift variable) is applied to construct the GSCVQ method. The proposed algorithm for selecting the best shift variables is performed and the minimum and maximum shift values for each central codeword are calculated. Figure 4 shows the histogram of these minimum and maximum shift values computed for the last subvector, i.e., LSF7-10.

In this experiment, the final selection of shift variables according to the new algorithm for the LSF7-10 subvector results in the histogram illustrated in Figure 5. It is obvious that the best value for the shift variable is not always zero. Hence, the proposed

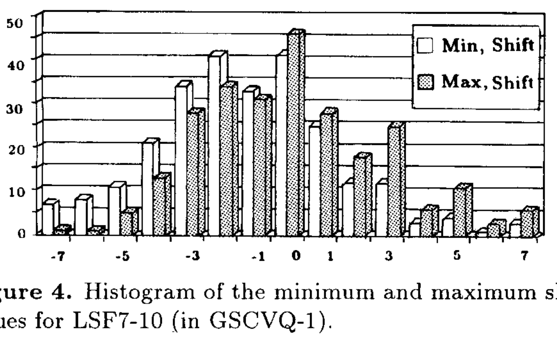


Figure 4. Histogram of the minimum and maximum shift values for LSF7-10 (in GSCVQ-1).

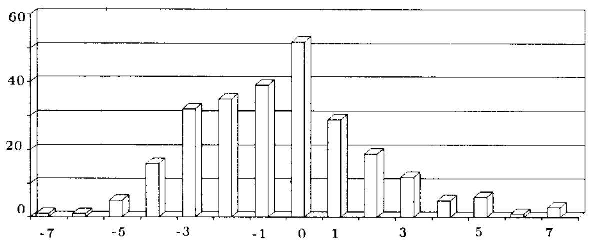


Figure 5. Histogram of the final shift values for LSF7-10 (in GSCVQ-1).

generalization approach can marginally reduce the quantization distortion. This was verified in this experiment, where the quantization of the entire training data set by this GSCVQ (GSCVQ-1) outperformed that of SCVQ (as will be shown shortly).

In the next experiment, the GSCVQ-1 is used along with another generalization of SCVQ that hereafter will be known as GSCVQ-2. This generalized SCVQ is developed by a combination of the second and third generalization approaches explained before, where the function set includes the sum of the elements in each subvector and the difference between the distance between the first two elements of each subvector and the distance between the second and third elements in that subvector.

In GSCVQ-2, the local search is conducted over 16 adjacent codevectors in the indices table (similar to that of SCVQ). All GSCVQ and SCVQ methods used the optimized codebooks.

The tests were conducted with 1,580 frames extracted from 18 utterances (47 seconds of speech) taken from TIMIT database with the same ratios of male/female speakers as the training database. Different male and female speakers were included in the training and test data sets. The LSFs of the test database were quantized by the various methods and were used in a simulated speech coder which is similar to FS1016 (apart from the LSF quantization method).

Results of simulations with unquantized LSFs and with the FS1016 quantization method (using 34 bits/frame) are also presented for comparison. Tables 1 and 2 depict the objective measures obtained via the various quantization schemes. In Table 1, SD denotes the log spectral distortion between the original and quantized LPC spectra.

In Table 2, the SSD is the synthesized spectral distortion that measures spectral distortion between the LPC spectra of the original signal and the one extracted from the synthesized speech [3]. The third objective measure is segmental signal to noise ratio (Seg-SNR), which evaluates the errors of the synthesized speech produced by the decoder.

The results presented in these tables demonstrate almost similar achievements for SCVQ and GSCVQ

Table 1. Results of spectral distortion comparison.

Quantization Method	SD [dB]	Percent of Frame with SD > 2dB	Percent of Frame with SD > 4dB
Unquantized	—	—	—
FS1016 Std.	1.53	12.34	0.57
UVQ	1.40	11.65	0.00
TSVQ	1.58	16.90	0.06
SCVQ	1.53	16.08	0.38
GSCVQ-1	1.53	16.07	0.32
GSCVQ-2	1.54	14.75	0.36

Table 2. Results of SSD and Seg-SNR comparisons.

Quantization Method	No. of Bits Per Frame	SSD [dB]	Seg-SNR [dB]
Unquantized	—	2.11	9.44
FS1016 Std.	34	2.30	8.77
UVQ	24	2.29	9.12
TSVQ	24	2.37	9.07
SCVQ	24	2.34	9.05
GSCVQ-1	24	2.34	9.03
GSCVQ-2	24	2.34	9.01

methods. For a more precise comparison, it is beneficial to compare the results of quantizations for VQ methods using a proportional MSE distortion measure as follows:

$$D = \left[\frac{\sum_{n=1}^{N} |\mathbf{F}_n - \hat{\mathbf{F}}_n|^2}{\sum_{n=1}^{N} |\mathbf{F}_n|^2} \right]^{1/2} \times 100 . \qquad (3)$$

where $\mathbf{F}_n$ and $\hat{\mathbf{F}}_n$ are the original and quantized vector in the nth frame, respectively, and N represents the total number of frames in the test data set. While this measure might not be as useful as the others for the spectral coding of speech, it is beneficial for representing the quantization errors in some other application areas.

Table 3 depicts the results of this distortion measure for different vector quantization methods applied to different LSF subvectors (D_1 through D_3 as well as the total distortion D).

In this example, it is obvious that GSCVQ-1 always results in lower distortion compared to SCVQ,

Table 3. Results of proportional MSE distortion.

Quantization Method	D_1	D_2	D_3	D
UVQ	3.48	2.06	1.48	1.69
TSVQ	4.07	2.38	1.68	1.93
SCVQ	3.91	2.23	1.72	1.92
GSCVQ-1	3.86	2.18	1.71	1.91
GSCVQ-2	3.87	2.34	1.66	1.90

Table 4. Cost comparison.

Quantization Method	Storage Cost [kbyte]	Computation Cost [No. of Instr.]
Unstructured VQ	5.00	7680
TSVQ	9.96	480
SCVQ	5.00	562
GSCVQ-1	5.75	565
GSCVQ-2	5.00	573

whereas for two out of three subvectors, the GSCVQ-2 provides lower distortion than SCVQ. In general, it can be stated that for any particular case, the statistics of the elements in each subvector throughout the training database indicate whether SCVQ or GSCVQ is a more suitable quantization method. In other words, the statistics of any target subvector indicate whether a one-dimensional or a multi-dimensional parameter is more appropriate to represent the neighborhood correlation between the elements of that target subvector.

Finally, the computational complexity and storage costs of different vector quantization methods are shown in Table 4.

The storage costs are calculated assuming single integer precision and shift values; double integer precision for the VQ elements and floating point for the scalar quantization elements. In the computational cost estimation, one instruction represents multiply-add comparison or data format conversion. It is seen that SCVQ and GSCVQ-2 result in the best storage cost compared to TSVQ. GSCVQ-1 is also preferred over TSVQ, because of its lower memory requirement. While the computational complexity of TSVQ is slightly lower than those of SCVQ and GSCVQ methods, the latter methods provide the best combinational complexity and storage costs.

As a final comment, it should be noted that not only the performances of the SCVQ and GSCVQ are directly related to the choice of the sorting function and the shift variables, but also the implementation costs of these methods are related to these important selections.

CONCLUSION

Different generalization aspects of the sorted codebook vector quantization have been presented. This includes changing the shift variable, neighborhood dependencies as well as using a multi-dimensional sorting function set. Moreover, a brief description of a method to select a proper sorting function and to optimize the codebook is presented. Application of a particular GSCVQ to spectral coding of speech is discussed as a case study and the results of experimental simulations are presented. As shown, the generalized sorted vector quantization is a promisingly efficient method for various applications.

REFERENCES

1. Finiceh, R., *Telecommunications: Analog to Digital Conversion of Radio Voice by 4,800 Bit/Second Code Excited Linear Prediction (CELP)*, Federal Standard 1016 National Communications Systems, Office of Technology and Standards, Washington, DC 20305-2010 (Feb. 14, 1991).
2. Itakura, F. "Line spectrum representation of linear predictive coefficients of speech signals", *J. Acoust. Soc. Am.*, **57**, p S35 (A) (1975).
3. Sadegh-Mohammadi, H.R. and Holmes, W.H. "Low cost vector quantization methods for spectral coding in low rate speech coders", *IEEE Proc. ICASSP*, **1**, pp 720-723 (May 1995).
4. Sadegh-Mohammadi, H.R. and Holmes, W.H. "Application of sorted codebook vector quantization to spectral coding of speech", *IEEE Proc. GLOBECOM*, **3**, pp 1595-1598 (Nov. 1995).
5. Sadegh-Mohammadi, H.R. and Holmes, W.H. "Efficient coding of line spectral frequencies using sorted codebook vector quantization", presented at *International Symposium on Information Theory and Its Applications, ISITA '96* (1996).
6. Makoul, J., Roucos, S. and Gish, H. "Vector quantization in speech coding", *Proc. IEEE*, **73**, pp 1551-1558 (Nov. 1985).