Low-Complexity Wideband LSF Quantization Using Algebraic Trellis VQ

Abdellah KADDAI†(a), Student Member and Mohammed HALIMI††, Nonmember

SUMMARY In this paper an algebraic trellis vector quantization (ATVQ) that introduces algebraic codebooks into trellis coded vector quantization (TCVQ) structure is presented. Low encoding complexity and minimum memory storage requirements are achieved using the proposed approach. It exploits advantages of both the TCVQ and the algebraic codebooks to know the delayed decision, the codebook widening, the low computational complexity and the no storage of codebook. This novel vector quantization scheme is used to encode the wideband speech line spectral frequencies (LSF) parameters. Experimental results on wideband speech have shown that ATVQ yields the same performance as the traditional split vector quantization (SVQ) and the TCVQ in terms of spectral distortion (SD). It can achieve a transparent quality at 47 bits/frame with a considerable reduction of memory storage and computation complexity when compared to SVQ and TCVQ.

key words: trellis coded quantization, algebraic codebook, LSF, wideband speech coding

1. Introduction

Many modern speech coding algorithms use the linear predictive coding (LPC) coefficients for describing and coding the speech spectral envelope. Each set of these coefficients is obtained from the input signal through linear prediction analysis of each frame of speech and then quantized for transmission. The LPC coefficients are usually transformed into some other representation more suitable for efficient quantization. The most interesting representation of the LPC coefficients is the line spectral frequencies (LSF) [1], [2].

The goal in the quantization of the LSF parameters is to obtain the transparent quality [3], at low bit-rate requirement with a minimum of memory storage and an acceptable level of computational complexity. Various quantization schemes have been suggested for the quantization of LSF parameters. Unstructured vector quantizers can perform well but with more complexity of codebook search and higher codebook storage [3]. Many authors have tried to find efficient quantizers that reduce the complexity and the memory requirements at the expense of degraded performance. In [4], Paliwal and Atal introduced the split vector quantization (SVQ) for narrowband speech LSF coding. The basic idea of SVQ is to split a high dimension vector into lower dimension vectors, which are then quantized separately. Typically, an individual quantizer is designed and used for each sub-vector. The multistage vector quantization of the LSF parameters is suggested in [5]. Another efficient way with a small complexity is the algebraic vector quantization proposed by Xie and Adoul [6]. It requires no storage and few operations compared to unstructured vector quantizers. More recently, many papers have reported on vector quantization schemes of wideband speech LSF parameters. The extended version of SVQ to the wideband case is proposed in [7]. In [8], a lattice vector quantization based on generalized voronoi codes is presented. So and Paliwal introduced switched SVQ (SSVQ) technique [9], which has shown better performance than SVQ at the expense of higher memory storage. In [10], a conditional PDF-based split vector quantization is proposed.

The duality between the modulation and the source coding as well as the alphabet constrained rate-distortion theory allowed to Fischer and Marcellin to suggest the trellis coded quantization (TCQ) [11] which uses Ungerboeck’s amplitude modulation trellises and set partitioning ideas of trellis coded modulation [12]–[14] to design coders for memoryless sources. This technique achieved good performance which is often better compared to the other methods of quantization in terms of both distortion and complexity. By exploiting the superior performance of vector quantization over scalar quantization, TCQ was extended to trellis coded vector quantization (TCVQ) [15] for further performance improvement. This, however, comes at the cost of an increase in memory and complexity requirements due to stochastic codebook storage and searching. Several studies on trellis coded quantization and its applications in speech coding field can be found in the literature. A trellis search inspired from TCQ and implemented into the ACELP fixed codebook is proposed in [16]. In [17], the TCVQ is used to encode the narrowband speech LSF parameters.

In this paper, we propose a solution to the problem of high memory and complexity requirements of TCVQ scheme mentioned above. The main contribution of the present paper is the development of a new approach called algebraic trellis vector quantization (ATVQ) by introducing algebraic codebooks into TCVQ in order to reduce computational complexity and memory storage requirements. ATVQ is designed for encoding wideband speech LSF parameters where the performance/complexity advantage is better than the narrowband case due to the high LPC orders and bit allocations utilized.
The remainder of the paper is organized as follows. Section 2 includes a brief overview of the properties of LSF parameters and describes the average spectral distortion and the distance measure utilized in this work. In Sect. 3, we present briefly the TCVQ. Section 4 provides the proposed ATVQ scheme. The numerical simulation results and comparisons are presented in Sect. 5. Our conclusions are given in Sect. 6.

2. Background

2.1 Line Spectral Frequencies

For the \( p \)-th order LPC analysis, an all-pole synthesis digital filter is given by:

\[ H(z) = \frac{1}{A(z)} = \frac{1}{1 + a_1 z^{-1} + \cdots + a_p z^{-p}} \]  

where \( \{a_i\}_{i=1}^p \) are the LPC parameters. The inverse filter polynomial \( A(z) \) is used to define two polynomials,

\[ P(z) = A(z) - z^{-(p+1)}A(z^{-1}) \]  

and

\[ Q(z) = A(z) + z^{-(p+1)}A(z^{-1}) \]

The polynomials \( P(z) \) and \( Q(z) \) possess three very interesting properties [2] summarized as:

1. All zeros of \( P(z) \) and \( Q(z) \) are on the unit circle.
2. The zeros of \( P(z) \) and \( Q(z) \) alternate each other on the unit circle.
3. Minimum phase property of \( A(z) \) is easily preserved after quantizing the zeros of \( P(z) \) and \( Q(z) \).

Since the zeros of \( P(z) \) and \( Q(z) \) are on the unit circle, they are given by \( e^{j\omega_i} \). The parameters \( \omega_i \), \( i = 1, \ldots, p \), are defined as LSF. From the second property given above, these parameters are ordered as

\[ 0 < \omega_1 < \omega_2 < \cdots < \omega_{p-1} < \omega_p < \pi \]

This relation is known as the ordering property of the LSF parameters. While quantizing LSF parameters, the stability of the synthesis filter \( H(z) \) is insured when the ordering property is preserved [3]. For computing LSF parameters, there are several methods. Among these, a method based on Chebyshev polynomials has been presented in [18].

2.2 Average Spectral Distortion

In order to evaluate the LPC quantizer performance, the average spectral distortion (SD) is often used, and is defined as follows

\[ SD = \frac{1}{N_f} \sum_{n=1}^{N_f} \left[ \frac{100}{\pi} \int_0^\pi \log_{10} \left( \frac{A_n(e^{j\omega})}{\hat{A}_n(e^{j\omega})} \right)^2 d\omega \right] \]

where SD is given in decibels, \( N_f \) is total number of frames, and \( A_n(e^{j\omega}) \) and \( \hat{A}_n(e^{j\omega}) \) are the original and quantized spectra of the \( n \)-th speech frame, respectively. The SD measure is known to have a good correspondence with subjective measures. In the case of narrowband LPC parameter quantization, it is commonly assumed that the quantization process is transparent if the following conditions are fulfilled [4]:

1. The average spectral distortion is approximately 1 dB.
2. Less than 2% of outlier frames should be within the range 2–4 dB.
3. No outliers should exist with SD larger than 4 dB.

The listening tests of Guibé et al. [19], have shown that the existing requirements for transparency in narrowband LPC coding also apply to the wideband case.

In practice, the spectral distortion is often replaced by a weighted Euclidean distance both in the designing and coding phases. The weighted distance usually improve the correspondence with SD and hence have a smaller SD than the unweighted distance at the same bit rates. The most well-known form of weighted distance is given by

\[ d(\omega, \hat{\omega}) = \sum_{i=1}^{p} w_i (\omega_i - \hat{\omega}_i)^2 \]

where \( \omega_i \) and \( \hat{\omega}_i \) are the \( i \)-th LSF parameter in the original and quantized vector respectively, and the vector \( w = [w_1, w_2, \ldots, w_p] \) is a variable weight, which is derived from the LSF parameters in each frame. Different weights have been proposed in the literature of quantization, all determined in an empirical way. Among these, a weighting function, called the inverse harmonic mean (IHM) weighting function, is introduced in [20] in order to exploit the local spectral variation resulting from perturbing any LSF parameter, and is defined as

\[ w_i = \frac{1}{\omega_i - \omega_{i-1}} + \frac{1}{\omega_{i+1} - \omega_i}, \quad i = 1, \ldots, p \]

where \( \omega_0 = 0, \omega_{p+1} = \pi \).

3. Trellis Coded Vector Quantization Overview

TCQ is a form of trellis coding that labels each branch of a trellis by a subset of the reconstruction levels [11]. The vector generalization of TCQ is quite similar in structure to the scalar TCQ. The trellis used in TCVQ can be any one of Ungerboeck’s trellis coded modulation trellises [12]. The TCVQ is based on a 1/2 binary convolutional encoder, which has \( N \) states with two branches leaving and entering each trellis state. For encoding \( L \)-dimensional source vectors at an encoding rate of \( R \) bits/sample, the expanded codebook used in TCVQ has \( 2^{R(L+1)} \) codevectors (i.e., \( 2^{RL} \) times as many as classical VQ). We refer to \( R \) as the codebook expansion factor (in bits per sample). According the Ungerboeck’s set partitioning rules [12], the expanded codebook is partitioned into \( M \) subsets \( \{M = 2^{RL-1}\} \), which each subset containing exactly \( 2^{RL-1} \) codevectors. Denote the
subsets $S_i$, $i = 0, 1, \ldots, M - 1$, such that $S_i$ and $S_{i+M/2}$ are partitions of the same intermediate subset in the last partitioning step [21]. These subsets is used for labeling the trellis branches.

The $n$ bits available for encoding each input vector are used as follows. One bit, called the transition bit, is used to determine the state transition represented by a trellis branch whereas the $n - 1$ other bits, called selection bits, are used to select a codevector within a given subset.

The encoding is accomplished in two steps [15].

**Step 1)** For each input, find the closest codevector and corresponding distortion in each subset.

**Step 2)** Let the branch metric for a branch labeled with subset $S$ be the distortion found in **Step 1**, and use the Viterbi algorithm [22] to find the minimum distortion path through the trellis.

The encoding process allows the winning trellis path to begin in any of $N$ initial states. Therefore, the bit sequence at the TCVQ quantizer output indicates the choice among the possible branches at each state plus a supplementary $v$ bits ($N = 2^v$), which specify the initial trellis state.

The design of TCVQ quantizer consists of several steps. These steps include selection of a trellis, construction of the initial expanded codebook, partitioning of this codebook into subsets, and the labeling of the trellis branches with these subsets.

### 4. Algebraic Trellis Vector Quantization

The basic idea of ATVQ is to substitute the stochastic codebook in TCVQ structure by Xie and Adoul’s algebraic codebook, in order to reduce the complexity and memory requirements. In this section, the proposed scheme is presented in wideband LSF parameter quantization context.

#### 4.1 ATVQ Structure

By inspiring from Xie and Adoul’s idea, the 16-dimensional wideband LSF vector is split into five subvectors as follows:

$$
\begin{align*}
\{ (\omega_0, \omega_4, \omega_8, \omega_{12}), (\omega_1, \omega_5, \omega_3), (\omega_2, \omega_6, \omega_7), \\
(\omega_9, \omega_{10}, \omega_{11}), (\omega_{13}, \omega_{14}, \omega_{15}) \}
\end{align*}
$$

(8)

It should be mentioned that the first subvector components correspond to the borders of the other subvectors according to the ordering property of the LSF parameters.

Each subvector is associated to a trellis stage. As a result, there are five stages in this trellis. The first trellis stage used a stochastic expanded codebook, which is partitioned into 8 or 16 subsets, whereas the last four stages utilized an algebraic expanded codebook partitioned into 16 subsets. For the encoding process, an 16- or 32-state trellis are used in all the cases considered in this work. Tables 1 and 2 list, respectively, a 16- and 32-state trellis populated with 8 and 16 subsets.

To design the stochastic expanded codebook from a training set of the LSF vectors $(\omega_0, \omega_4, \omega_8, \omega_{12})$, the generalized Lloyd algorithm (GLA) [23] can be employed. The choice of the initial codebook for GLA algorithm has also some importance. The algorithm developed in [24] can be used to built a better initial codebook required by GLA algo-

---

**Table 1**  Trellis structure and branch labeling for a 16-state trellis populated with 8 and 16 subsets.

| Current State | Previous State | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---------------|----------------|---|---|---|---|---|---|---|---|
| Pre. State    |                | 0 | 2 | 4 | 6 | 8 | 10| 12|14 |
| 8-Ass. Subsets|                | 4 | 6 | 0 | 2 | 12|14 | 8 |10 |
| 16-Ass. Subsets|               | 0 | 4 | 2 | 6 | 0 | 4 | 2 | 6 |
| Current State |                | 8 |12 |10|14 | 8 |12 |10|14 |

**Table 2**  Trellis structure and branch labeling for a 32-state trellis populated with 8 and 16 subsets.

| Current State | Previous State | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---------------|----------------|---|---|---|---|---|---|---|---|
| Pre. State    |                | 0 | 2 | 4 | 6 | 8 | 10| 12|14 |
| 8-Ass. Subsets|                | 4 | 6 | 0 | 2 | 12|14 | 8 |10 |
| 16-Ass. Subsets|               | 0 | 4 | 2 | 6 | 0 | 4 | 2 | 6 |
| Current State |                | 8 |12 |10|14 | 8 |12 |10|14 |

---

IEICE TRANS. INF. & SYST., VOL.E92-D, NO.12 DECEMBER 2009
rithm. Once the stochastic expanded codebook is selected, a codebook partitioning into subsets is the next step, which can be accomplished by using the method proposed in [25].

The last four subvectors have undertaken some transformations for the following reasons.

1. To exploit the intraframe correlation between two neighboring LSF parameters.
2. To adapt the distribution of the subvectors to be quantized with algebraic codebooks structure.
3. To reduce the global distortion of quantization.
4. To avoid violation of the ordering property of the LSF parameters after quantization.

Consequently, after application of the transformations, the last four subvectors become \((x_1, x_2, x_3, x_4)\) and defined as follows.

\[
\begin{align*}
    x_{11} &= \frac{\omega_1 - \hat{\omega}_0}{\hat{\omega}_4 - \hat{\omega}_0} \quad x_{12} = \frac{\omega_2 - \hat{\omega}_1}{\hat{\omega}_4 - \hat{\omega}_0} \quad x_{13} = \frac{\omega_3 - \hat{\omega}_2}{\hat{\omega}_4 - \hat{\omega}_0} \\
    x_{21} &= \frac{\omega_3 - \hat{\omega}_4}{\hat{\omega}_8 - \hat{\omega}_4} \quad x_{22} = \frac{\omega_4 - \hat{\omega}_5}{\hat{\omega}_8 - \hat{\omega}_4} \quad x_{23} = \frac{\omega_5 - \hat{\omega}_6}{\hat{\omega}_8 - \hat{\omega}_4} \\
    x_{31} &= \frac{\omega_6 - \hat{\omega}_8}{\hat{\omega}_{10} - \hat{\omega}_8} \quad x_{32} = \frac{\omega_8 - \hat{\omega}_9}{\hat{\omega}_{10} - \hat{\omega}_8} \quad x_{33} = \frac{\omega_9 - \hat{\omega}_{10}}{\hat{\omega}_{10} - \hat{\omega}_8} \\
    x_{41} &= \frac{\omega_{13} - \hat{\omega}_{12}}{\mu - \hat{\omega}_{12}} \quad x_{42} = \frac{\omega_{14} - \hat{\omega}_{13}}{\mu - \hat{\omega}_{12}} \quad x_{43} = \frac{\omega_{15} - \hat{\omega}_{14}}{\mu - \hat{\omega}_{12}}
\end{align*}
\]

where \(\omega_i\) and \(\hat{\omega}_i\) are the \(i\)th LSF parameter in the original and quantized vector respectively. The constant \(\mu\) used in Eq. (12), in the place of \(\pi\) as in [6], is equal to 3.0245. It is defined as the maximum value of the 16-th LSF. Our experiments indicate the effectiveness of \(\mu\) value than \(\pi\) in terms of quantization performances. Note that for determining the value of \(\mu\) from the training database, the histogram of the 16-th LSF is taken into account where the very large values are discarded (one percent of the total number).

When we substitute the quantized values in Eqs. (9)–(12) by their original (unquantized) values, we can depict the scatter diagrams of the components of subvectors \(x_1, x_2, x_3\) and \(x_4\) (Fig. 1).

It can be seen from Fig. 1 that the joint distribution of the components of each subvector has a pyramid shape \((i.e.\ a\ regular\ form)\). It is very interesting to benefit from this geometric property by using, in quantization process, an algebraic codebook \((i.e.\ Lattice: a\ regular\ set\ of\ points\ in\ space)\), which has the two following attractive features:

- The lattice structure allows implementing rapid algorithms to find the closest lattice point to a given source vector \(x\).
- The codevectors are generated in a simple manner rather than stored in a codebook.

We now focus our attention on the structure of the algebraic expanded codebook as well as its design procedure.

4.2 Algebraic Expanded Codebook

The subsets which constitute the algebraic expanded codebook are produced based on lattice \(D_3\), which is defined as

\[
D_3 = \left\{ c \in \mathbb{Z} | \sum_{i=1}^{3} c_i \text{ is an even integer} \right\}
\]

In practical applications of lattice quantization, the lattices are usually truncated. For minimum distortion, the ideal truncation shape is a contour of constant probability density for the considered source as mentioned in [26]. In our case, lattice \(D_3\) is truncated according to the pyramid form by employing the constraints expressed as follows.

\[
c_i \geq 0, \quad i = 1, 2, 3 \quad \text{and} \quad \sum_{i=1}^{3} c_i \leq N
\]

where \(c_i\) is the \(i\)th component of a codevector in the lattice codebook and \(N\) is a fixed even positive integer which defines the size of codebook. Table 3 presents for each number of bits, the corresponding value of \(N\) as well as the size of algebraic codebook.

As we have mentioned above, the algebraic expanded codebook is formed by 16 subsets. While referring to Tables 1 and 2, and following the ad hoc design rules described in [15] for subset construction and trellis branch labeling, the subsets \(S_i, i = 0, 1, \ldots, 15\), are expressed as follows.

![Fig. 1 Scatter diagrams of the components of subvectors \(x_1, x_2, x_3\) and \(x_4\).](image)

| Number of Bits | \(N\) | Codebook Size |
|---------------|------|--------------|
| 4             | 4    | 10           |
| 5             | 6    | 20           |
| 6             | 10   | 56           |
| 7             | 14   | 120          |
| 8             | 18   | 220          |
| 9             | 24   | 455          |
| 10            | 32   | 969          |

![Table 3 Algebraic codebook size for different value of \(N\).](image)
Table 4  List of the translation vectors corresponding to the subsets.

| Subset | Translation vector |
|--------|--------------------|
| $S_0$ | (0,0,0)            |
| $S_1$ | (0,0,1)            |
| $S_2$ | (0,1,1)            |
| $S_3$ | (0,1,0)            |
| $S_4$ | (1,1,1)            |
| $S_5$ | (1,1,0)            |
| $S_6$ | (1,0,1)            |
| $S_7$ | (1,0,0)            |
| $S_8$ | (1,1,1)            |
| $S_9$ | (1,1,0)            |
| $S_{10}$ | (1,0,0)     |
| $S_{11}$ | (1,0,1)      |
| $S_{12}$ | (0,0,0)     |
| $S_{13}$ | (0,0,1)     |
| $S_{14}$ | (0,1,1)     |
| $S_{15}$ | (0,1,0)     |

$S_i = \{D_3 + v_i\}, \ i = 0, 1, \ldots, 15$  \hspace{1cm} (15)

where $\{D_3 + v_i\}$ is a notation for the set containing the vectors of the lattice $D_3$ translated by the vector $v_i$. Table 4 shows the corresponding vector $v_i$ for each subset $S_i$.

Note that, from Table 4, there are some identical subsets. For example, $S_0$ and $S_{12}$.

With an aim to achieve high quantizer performance, different scale factors $\alpha_i$, $i = 0, 1, \ldots, 7$, are assigned to the 8 nonidentical subsets. These scale factors must be suited to the joint distribution of the subvector components. In purpose to take advantage of the first feature of lattice codebook cited previously, the input vector components are scaled with these factors $\alpha_i$ rather than the lattice codevector components. Before scaling step, the input vector is translated in the negative sense by an offset vector $\beta = (\beta_x, \beta_y, \beta_z)$. This approach has the advantage of providing more performance improvement.

An effective way to find good scale factors is accomplished by training. For each subset at a given rate, we try various scale factors, evaluate the mean squared error of each after encoding the training sequence using the algebraic subset, and built up a list of best scale factors.

The scaling issue of the identical subsets, noted $S_i$ and $S_j$ with $i < j$, is given in the following method. We assign $\alpha_i$ to $S_i$ and $(\gamma \alpha_i)$ to $S_j$, where $\gamma$ is a scalar factor obtained by training with the same manner as $\alpha_i$.

The offset vector components $(\beta_x, \beta_y, \beta_z)$ correspond to the pyramid origin point, which can be determined empirically by discarding a portion of the pyramid shape of the joint distribution of subvector $x_i$ from its inferior part in the three space directions. A portion equal to five percent (5%) has been found satisfactory.

It should be noted that we follow, in our work, the design procedure explained above to build all of the four algebraic expanded codebooks corresponding to the last four trellis stages.

4.3 Nearest Neighbor Search and Indexing

The next step after codebook building is encoding, which can be divided into two important parts. The first is nearest neighbor search, that is to find the closest codevector to a given input vector. The second is indexing, that is to obtain the index of a given codevector.

For a given codebook and an input vector, the optimal nearest neighbor search (full search version) is to compare the input vector to each codevector in the codebook and thus finding the codevector that minimizes the distortion criterion. It can be applied for codebooks with small size, but it becomes impractical when increasing the size of codebook. This approach may be time consuming and cause large delays. The lattice codebook is one of the best solutions for reducing this delay. Conway and Sloane have developed fast quantization algorithms for a certain class of lattice quantizers [27]. The lattice $D_3$ we use in this work fall into that class. The nearest neighbor search inside lattice $D_3$ is carried out as follows:

**Step 1)** Find, by rounding, the closest codevector $y$ of the $Z^3$ lattice to an input vector $x$.

**Step 2)** If the sum of components of $y$ is odd. Modify $y$ by rounding in bad way the component of $x$ which has a greater distortion of quantization (i.e. rounding) than the others.

In this paper, the nearest neighbor search in each subset is accomplished in very simpler manner according to its corresponding translation vector. Let $a = (a_1, a_2, a_3)$ be any input vector of three dimensions, and let $v_i = (v_{i1}, v_{i2}, v_{i3})$ be the translation vector associated to the subset $S_i$. Rounding the input vector component $a_j$, $j = 1, 2, 3$, to the nearest even integer if $v_{ij} = 0$, and to the nearest odd integer if $v_{ij} = 1$.

In this encoding phase, we must devise an efficient algorithm to resolve the problem of the input vectors which fall outside the truncated lattice region. In literature, these input vectors are called overload vectors. In order to solve this problem, we have developed an algorithm which is given in detail in the following.

**Algorithm for Overload Vector Quantization:** Let $a = (a_1, a_2, a_3)$ be an overload vector and $N$ an even integer number (defined in Sect. 4.2). Denoting the sum of each translation vector $v_i$ (corresponding to subset $S_i$) as $m_i$.

1) Find the closest codevector, $c = (c_1, c_2, c_3)$, to overload vector $a$ in subset $S_i$.

2) If this codevector lies outside the truncated lattice $\{i.e. \sum_{j=1}^{3} c_j > (N + m_i)\}$, then calculate the difference between $c$ and $a$. This difference vector $d$ is designated as $d_i = c_i - a_i$, $i = 1, 2, 3$.

3) Create an index vector, $id = (id_1, id_2, id_3)$, for the vector $d$, i.e. a vector of pointers telling which $d_i$, $i = 1, 2, 3$, comes first in decreasing order, which second, and so on.

4) Set $c(id_1)$ to $c(id_1) - 2$. If $\sum_{j=1}^{3} c_j \leq (N + m_i)$, then stop.

Otherwise continue.
5) Set \( c(id_2) \) to \( c(id_2) - 2 \). If \( \sum_{j=1}^{3} c_j \leq (N + m) \), then stop.
Otherwise continue.
6) Set \( c(id_3) \) to \( c(id_3) - 2 \). If \( \sum_{j=1}^{3} c_j \leq (N + m) \), then stop.
Otherwise return to Step 4.

For evaluation, computer simulations have been carried out on this algorithm and on the full search algorithm. The results show that both algorithms provide, for an input overload vector, the same closest codevector in truncated lattice.

To use lattice codebooks in a particular application, we must build up an indexing function (enumeration encoding), which produces an index for each codevector of the truncated lattice, and also its inverse function (enumeration decoding), which assigns to each index the corresponding codevector of the truncated lattice. Based on the indexing method presented in [6], we have proposed a modified version of this method. Its details are summarized as follows.

**Encoding Algorithm**: Let \( c = (c_1, c_2, c_3) \) be a codevector of the truncated lattice, and let \( v_i = (v_{i1}, v_{i2}, v_{i3}) \) be the translation vector associated to the subset \( S_i \).

1) Compute
   \[
   t_j = \left( c_j - v_{ij} \right)/2, \quad \text{for } j = 1, 2, 3.
   \]
   \( s_2 = t_1 + t_2, \)
   \( s_3 = s_2 + t_3, \)
   \( k_2 = \frac{1}{3}s_2 (s_2 + 1). \)
   \( k_3 = \frac{1}{3}s_3 (s_3 + 1) (s_3 + 2). \)
2) The index to be transmitted is: \( k = k_3 + k_2 + t_1. \)

Note that the value of \( k_3 \) represent the number of codevectors in the pyramid shape (three dimensions) which their sum is lower than \((2s_3)\). The possible values of \( k_3 \) are: \{0, 1, 4, 10, 20, 35, 56, 84, 120, 165, \ldots \} corresponding to the following values of \( s_3: \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, \ldots \} \). The value \( k_2 \) has the same significance of \( k_3 \) but in the triangular shape (two dimensions). The possible values of \( k_2 \) are: \{0, 1, 3, 6, 10, 15, 21, 28, 36, 45, \ldots \} corresponding to the following values of \( s_2: \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, \ldots \} \).

**Decoding Algorithm**: Let \( v_i = (v_{i1}, v_{i2}, v_{i3}) \) be the translation vector associated to the subset \( S_i \). Given an index \( k \)

1) Extract the value of \( k_3 \) from \( k \) where \( k_3 \) is the greater value, within the list of its possible values, lower or equal to \( k \). Denote the position of the found value of \( k_3 \) within the list of its possible values as \( l \), where the first value is considered at position 0.
2) Extract the value of \( k_2 \) from \( k - k_3 \) where \( k_2 \) is the greater value, within the list of its possible values, lower or equal to \((k - k_3)\). Denote the position of the found value of \( k_2 \) within the list of its possible values as \( m \).
3) Compute \( t_1 = (k - k_3 - k_2) \).
4) The codevector \( c \) corresponding to the index \( k \) is given by their components as:
   \( c_1 = v_{i1} + 2t_1; \)
   \( c_2 = v_{i2} + 2(m - t_1); \)
   \( c_3 = v_{i3} + 2(l - m); \)

5. **Simulation Results**

The speech material used in this work consisted of 233007 frames of the TIMIT database [28] (with sampling rate 16 kHz) for training and testing phase. From this number, 121478 frames from 80 speakers were used for training codebooks, and 111529 frames from 73 speakers were reserved for evaluation. All speech files are filtered using P341 mask. The input speech level is adjusted to \(-26\) dBov. We have used the preprocessing and LPC analysis of the AMR-WB speech codec (floating point version) [29] to produce 16th order linear prediction coefficients which are then converted to LSF parameters (i.e. \( p = 16\)).

The bit allocation used in our experiments within ATVQ structure, that yielded the best performance is given in Table 5. The number of subsets used in each trellis stage at bit rates from 43 to 47 bits/frame are summarized in Table 6. The Table 7 lists the SD performance of the ATVQ and its ROM requirements at different bit rates. We can see from this table that ATVQ requires 47 bits/frame to achieve transparent quality LSF quantization.

We consider two schemes for comparison with our proposed ATVQ scheme. The traditional SVQ is used as reference for performances comparison such as the work presented in [9] and [10]. We have also simulated the TCVQ proposed by Fischer et al. presented in [15]. For both quantization schemes we consider different bit-rates from 43 to 47 bits/frame.

**Table 5** Bit allocation to each trellis stage of the algebraic trellis vector quantization as a function of bit-rate.

| Rate (bits/frame) | Initial State | Path | 1st stage | 2nd stage | 3rd stage | 4th stage | 5th stage |
|-------------------|--------------|------|-----------|-----------|-----------|-----------|-----------|
| 43                | 4            | 5    | 5         | 5         | 5         | 5         | 5         |
| 44                | 5            | 5    | 5         | 5         | 5         | 5         | 5         |
| 45                | 7            | 7    | 7         | 7         | 7         | 7         | 7         |
| 46                | 7            | 7    | 7         | 7         | 7         | 7         | 7         |
| 47                | 6            | 6    | 7         | 7         | 7         | 7         | 7         |

**Table 6** Number of subsets in each trellis stage of the algebraic trellis vector quantization for different bit-rates.

| Number of subsets in stages | Rate (bits/frame) | 43 | 44 | 45 | 46 | 47 |
|-----------------------------|-------------------|----|----|----|----|----|
| 1st stage                   |                   | 16 | 16 | 16 | 8  | 8  |
| Number of subsets in stages (2,3,4,5) |                | 16 | 16 | 16 | 16 | 16 |

**Table 7** Average spectral distortion (SD) performance, and memory requirements (ROM) of the algebraic trellis vector quantization as a function of bitrate on wideband LSF vectors from TIMIT database.

| Rate (bits/frame) | SD (dB) | Outliers (%) | ROM (floats) |
|-------------------|---------|--------------|--------------|
| 43                | 1.16    | 0.35         | 0.00         | 8192         |
| 44                | 1.12    | 0.28         | 0.00         | 8192         |
| 45                | 1.08    | 0.24         | 0.00         | 8192         |
| 46                | 1.04    | 0.21         | 0.00         | 8192         |
| 47                | 1.01    | 0.19         | 0.00         | 8192         |
Table 8  Average spectral distortion (SD) performance, and memory requirements (ROM) of the five part split vector quantization as a function of bitrate on wideband LSF vectors from TIMIT database.

| Rate (bits/frame) | Rate (bits/frame) | Rate (bits/frame) | Rate (bits/frame) | Rate (bits/frame) | Rate (bits/frame) | Rate (bits/frame) | Rate (bits/frame) |
|-------------------|-------------------|-------------------|-------------------|-------------------|-------------------|-------------------|-------------------|
| 43 (8,9,9,8,9)    | 44 (8,9,9,9,9)    | 45 (9,9,9,9,9)    | 46 (9,9,9,9,10)   | 47 (9,9,10,9,10)  |
| SD (dB)           | SD (dB)           | SD (dB)           | SD (dB)           | SD (dB)           |
| 1.16              | 1.12              | 1.10              | 1.05              | 1.02              |
| 1.06              | 0.75              | 0.66              | 0.40              | 0.28              |
| 4dB               | 4dB               | 4dB               | 4dB               | 4dB               |
| 0.00              | 0.00              | 0.00              | 0.00              | 0.00              |
| ROM (floats)      | ROM (floats)      | ROM (floats)      | ROM (floats)      | ROM (floats)      |
| 6656              | 7324              | 8192              | 10240             | 11776             |
| > 4dB             | > 4dB             | > 4dB             | > 4dB             | > 4dB             |
| 7424              | 10240             | 11776             | 14848             | 19968             |

Table 9  Bit allocation to each trellis stage of the trellis coded vector quantization as a function of bit-rate.

| Rate (bits/frame) | Rate (bits/frame) | Rate (bits/frame) | Rate (bits/frame) | Rate (bits/frame) |
|-------------------|-------------------|-------------------|-------------------|-------------------|
| 43                | 44                | 45                | 46                | 47                |
| Initial State     | 5                 | 5                 | 5                 | 5                 |
| Path              | 5                 | 5                 | 5                 | 5                 |
| 1st stage         | 6                 | 6                 | 7                 | 7                 |
| 2nd stage         | 7                 | 7                 | 7                 | 7                 |
| 3rd stage         | 7                 | 7                 | 7                 | 8                 |
| 4th stage         | 6                 | 7                 | 7                 | 7                 |
| 5th stage         | 7                 | 7                 | 8                 | 8                 |

Table 10  Average spectral distortion (SD) performance, and memory requirements (ROM) of the trellis coded vector quantization as a function of bitrate on wideband LSF vectors from TIMIT database.

| Rate (bits/frame) | Rate (bits/frame) | Rate (bits/frame) | Rate (bits/frame) | Rate (bits/frame) |
|-------------------|-------------------|-------------------|-------------------|-------------------|
| 43                | 44                | 45                | 46                | 47                |
| SD (dB)           | SD (dB)           | SD (dB)           | SD (dB)           | SD (dB)           |
| 1.16              | 1.12              | 1.09              | 1.04              | 1.00              |
| 0.58              | 0.37              | 0.30              | 0.16              | 0.09              |
| 0.00              | 0.00              | 0.00              | 0.00              | 0.00              |
| ROM (floats)      | ROM (floats)      | ROM (floats)      | ROM (floats)      | ROM (floats)      |
| 13312             | 14848             | 16384             | 20480             | 23552             |
| > 4dB             | > 4dB             | > 4dB             | > 4dB             | > 4dB             |
| 23552             | 46256             | 13312             | 23552             | 46256             |

The performance of SVQ and its ROM requirement are summarized in Table 8. Also shown are the bit allocations used, which obtain a better quantization performance. The bit allocation and performance of TCVQ scheme, with 32-state trellis and five stages, are presented in Tables 9 and 10, respectively. In comparison to SVQ and TCVQ, we observe that the ATVQ scheme provides nearly same performance as a function of bitrate on wideband LSF vectors from TIMIT database.

Fig. 2  Spectral distortion (SD) histograms for the three schemes SVQ, TCVQ, and ATVQ at 47 bits/frame.

Table 11  Comparison of nearest neighbor search complexity for the three schemes SVQ, TCVQ, and ATVQ at the bit-rate 47 bits/frame.

| Operation        | SVQ     | TCVQ    | ATVQ    |
|------------------|---------|---------|---------|
| Multiplication   | 11776   | 23552   | 8960    |
| Addition         | 19968   | 40064   | 15744   |
| Comparison       | 3579    | 7314    | 2226    |
| Rounding         | 0.00    | 0.00    | 768     |

Table 12  Scale factors and components of offset vector used in algebraic trellis vector quantization at 47 bits/frame.

| Trellis stage | 2nd stage | 3rd stage | 4th stage | 5th stage |
|---------------|-----------|-----------|-----------|-----------|
| a_0           | 21.5      | 26.6      | 22.4      | 21.5      |
| a_1           | 22.7      | 27.8      | 23.9      | 23.0      |
| a_2           | 24.2      | 29.3      | 25.3      | 24.3      |
| a_3           | 22.8      | 27.9      | 23.9      | 23.0      |
| a_4           | 25.7      | 30.9      | 27.0      | 26.0      |
| a_5           | 24.3      | 29.4      | 25.5      | 24.6      |
| a_6           | 22.8      | 27.9      | 24.0      | 23.0      |
| a_7           | 24.2      | 29.4      | 25.4      | 24.3      |
| γ              | 0.93      | 0.94      | 0.92      | 0.92      |
| β_1            | 0.04      | 0.08      | 0.08      | 0.09      |
| β_2            | 0.10      | 0.07      | 0.08      | 0.10      |
| β_3            | 0.09      | 0.06      | 0.08      | 0.10      |
codebooks into TCVQ. Issues of design, nearest neighbor search, overload vector quantization and indexing as well as the performance and comparison with traditional SVQ and TCVQ methods are addressed. Numerical results demonstrate that the proposed ATVQ gives the same performance as SVQ and TCVQ but saves memory and reduces complexity. ATVQ requires 47 bits/frame to achieve the transparent quality quantization performance. It is worth noting that the proposed method delivers storage/complexity advantage over SVQ for the bit-rate greater than 45 bits/frame, and over TCVQ for all of the bit rates examined.

References

[1] F. Itakura, “Line spectrum representation of linear predictive coefficients of speech signals,” J. Acoust. Soc. Am., vol.57, Suppl.1, p.335(A), April 1975.
[2] F.K. Soong and B.H. Juang, “Line spectrum pair (LSP) and speech data compression,” Proc. IEEE Int. Conf. on Acoust. Speech Signal Processing, pp.10.1–10.4, March 1984.
[3] K.K. Paliwal and W.B. Kleijn, “Quantization of LPC parameters,” in Speech Coding and Synthesis, ed. W.B. Kleijn and K.K. Paliwal, pp.433–466, Elsevier Science, New York, 1995.
[4] K.K. Paliwal and B.S. Atal, “Efficient vector quantization of LPC parameters at 24 bits/frame,” IEEE Trans. Speech Audio Process., vol.1, no.1, pp.3–14, Jan. 1993.
[5] W.F. Leblanc, B. Bhattacharya, S.A. Mahmoud, and V. Cuperman, “Efficient search and design procedures for robust multi-stage VQ of LPC parameters for 4 kb/s speech coding,” IEEE Trans. Speech Audio Process., vol.1, no.4, pp.373–385, Oct. 1993.
[6] M. Xie and J.P. Adoul, “Algebraic vector quantization of LSF parameters with low storage and computational complexity,” IEEE Trans. Speech Audio Process., vol.4, no.3, pp.234–239, May 1996.
[7] G. Biundo, S. Grassi, M. Ansorge, F. Pellandini, and P.A. Farine, “Design techniques for spectral quantization in wideband speech coding,” Proc. 3rd COST 276 Workshop on Information and Knowledge Management for Integrated Media Communication, pp.114–119, Budapest, Oct. 2002.
[8] S. Ragot, H. Lahdili, and R. Lefebvre, “Wideband LSF quantization by generalized Voronoi codes,” Proc. Eurospeech, pp.2319–2322, Aalborg, Denmark, Sept. 2001.
[9] S. So and K.K. Paliwal, “A comparative study of LPC parameter representations and quantization schemes for wideband speech coding,” Digit. Signal Process., vol.17, no.1, pp.138–171, Jan. 2007.
[10] S. Chatterjee and T.V. Sreenivas, “Conditional PDF-based split vector quantization of wideband LSF parameters,” IEEE Signal Process. Lett., vol.14, no.9, pp.641–644, Sept. 2007.
[11] M.W. Marcellin and T.R. Fischer, “Trellis coded quantization of memoryless and Gauss-Markov sources,” IEEE Trans. Commun., vol.38, no.1, pp.82–93, Jan. 1990.
[12] G. Ungerboeck, “Channel coding with multilevel/phase signals,” IEEE Trans. Inf. Theory, vol.28, no.1, pp.55–67, Jan. 1982.
[13] G. Ungerboeck, “Trellis-coded modulation with redundant signal sets Part I: Introduction,” IEEE Commun. Mag., vol.25, pp.5–11, Feb. 1987.
[14] G. Ungerboeck, “Trellis-coded modulation with redundant signal sets Part II: State of the art,” IEEE Commun. Mag., vol.25, pp.12–21, Feb. 1987.
[15] T.R. Fischer, M.W. Marcellin, and M. Wang, “Trellis coded vector quantization,” IEEE Trans. Inf. Theory, vol.37, no.6, pp.1551–1566, Nov. 1991.
[16] M. Halimi, A. Kaddai, and M. Bengherabi, “A new multistage search of algebraic CELP codebooks based on trellis coding,” IEICE Trans. Inf. & Syst., vol.E86-D, no.3, pp.406–411, March 2003.
[17] J. Pan and T.R. Fischer, “Vector quantization of speech line spectrum pair parameters and reflection coefficients,” IEEE Trans. Speech Audio Process., vol.6, no.2, pp.106–115, March 1998.
[18] P. Kabal and R. Ramachandran, “The computation of line spectral frequencies using Chebyshev polynomials,” IEEE Trans. Acoust. Speech Signal Process., vol.34, no.6, pp.1419–1426, Dec. 1986.
[19] G. Guibert, H.T. How, and L. Hanzo, “Speech spectral quantizers for wideband speech coding,” European Trans. Telecommunications, vol.12, no.6, pp.535–545, 2001.
[20] R. Laroia, N. Phamdo, and N. Farvardin, “Robust and efficient quantization of speech LSP parameters using structured vector quantizers,” Proc. IEEE Int. Conf. on Acoust. Speech Signal Processing, pp.641–644, May 1991.
[21] K.T. Malone and T.R. Fischer, “Trellis-searched adaptive predictive coding of speech,” IEEE Trans. Speech Audio Process., vol.1, no.2, pp.196–206, April 1993.
[22] G.D. Forney, Jr., “The Viterbi algorithm,” Proc. IEEE, vol.61, pp.268–278, March 1973.
[23] Y. Linde, A. Buzo, and R.M. Gray, “An algorithm for vector quantizer design,” IEEE Trans. Commun., vol.28, no.1, pp.84–95, Jan. 1980.
[24] I. Katsavounidis, C.J. Kuo, and Z. Zhang, “A new initialization technique for generalized Lloyd iteration,” IEEE Signal Process. Lett., vol.1, pp.144–146, Oct. 1994.
[25] H.S. Wang and N. Mouayeri, “Trellis coded vector quantization,” IEEE Trans. Commun., vol.40, no.8, pp.1273–1276, Aug. 1992.
[26] D.G. Jeong and J.D. Gibson, “Uniform and piecewise uniform lattice vector quantization for memoryless Gaussian and Laplacian sources,” IEEE Trans. Inf. Theory, vol.39, no.3, pp.786–803, May 1993.
[27] J.H. Conway and N.J.A. Sloane, “Fast quantizing and decoding algorithms for lattice quantizers and codes,” IEEE Trans. Inf. Theory, vol.28, no.2, pp.227–232, March 1982.
[28] NIST, The DARPA TIMIT Acoustic-Phonetic Continuous Speech Corpus, Oct. 1990.
[29] “3rd generation partnership project; Technical specification group services and system aspects; Speech codec speech processing functions; Adaptive multi-rate - Wideband (AMR-WB) speech codec; Transcoding functions (Release 7),” 3GPP TS 26.190, June 2007.
[30] T. Eriksson, J. Linden, and J. Skoglund, “Interframe LSF quantization for noisy channels,” IEEE Trans. Speech Audio Process., vol.7, no.5, pp.495–509, Sept. 1999.

Abdellah Kaddai was born in El-Attaf, Algeria, on October 5, 1974. He received the State Engineer degree in Electronics from the Ecole Nationale Polytechnique d’Harrach, Algeria, in June 1999 and the Magister degree in Advanced Signal Processing from the Ecole Militaire Polytechnique de Bordj El-Bahri, Algeria, in December 2001. In March 2002, he joined the Coding Team, Centre de Developpement des Technologies Avancées (CDTA). Since January 2003 he has been with Department of Electronics, Hassiba Benboulou University at Chlef as Assistant Professor. He is currently working towards Doctorate’s degree at the Department of Electronics, Saad Dahlab University at Blida. His current interests are in the area of vector quantization and its applications in speech coding field.
Mohammed Halimi was born in Algiers, Algeria in 1958. He received the diploma of Ingénieur d’État en Electronique from the Ecole Nationale Polytechnique d’Alger, Algeria in 1981, the Diplôme d’Etudes Approfondies en Electronique from ENSEEIHT/Institut National Polytechnique de Toulouse, France in 1982 and the diploma of Docteur de l’Institut National Polytechnique de Toulouse en Electronique from France in 1985. He joined the Centre de Développement des Technologies Avancées in 1985 where he was the head of the speech coding team. In 2003, he joined the Centre de Recherche Scientifique et Technique en Contrôle et Soudage where he is currently the head of the image segmentation team. His current research activities are concentrated on low bit rate speech coding, edge detection and image processing.