Fine-Granular Scalable MELP Coder Based on Embedded Vector Quantization

Mouloud Djamah, Douglas O’Shaughnessy

INRS-EMT : 800, de la Gauchetière Ouest Bureau 6900, Montreal (Quebec) H5A 1K6 CANADA
djamah@emt.inrs.ca

Abstract
This paper presents an efficient codebook design for tree-structured vector quantization (TSVQ), which is embedded in nature. The federal standard MELP (mixed excitation linear prediction) speech coder is modified by replacing the original single stage vector quantizer for Fourier magnitudes with a TSVQ and the original multistage vector quantizer (MSVQ) for line spectral frequencies (LSF’s) with a multistage TSVQ (MTVQ). The modified coder is fine-granular bit-rate scalable with gradual change in quality for the synthetic speech when the number of bits available for LSF and Fourier magnitudes decoding is decremented bit-by-bit.

Index Terms: Speech coding, MELP coder, tree-structured vector quantization, embedded quantization, scalability.

1. Introduction
In many standard coders, the encoder generates only one type of bit-stream at a fixed rate [1]. However, if the traffic in the transmission channel (packet network) is congested, the encoded data could be lost. This problem can be solved by using a scalable bit-stream consisting of a base layer followed by one or several enhancement layers. The enhancement layers are used to improve (from the base layer) the quality and they can be discarded one layer at time when the channel is congested. Fine Granularity Scalability [2] is an approach wherein the bit-stream can be discarded with finer granularity, on a bit-by-bit basis in the extreme case, instead of a whole layer. The need for scalable coders is largely due to the deployment of speech transmission over packet networks. The ITU-T G.729.1 [3] is a scalable speech coder that has been recently standardized. In general speech coders that incorporate embedded quantization are scalable. A quantizer is embedded when the quantized parameter can be successively refined as the associated index is read. In this work we modify the MELP coder by replacing two original quantizers (for LSF and Fourier magnitude quantization) by two TSVQ-based quantizers which are embedded in nature. The modified coder is fine-granular bit-rate scalable. The rest of the paper is organized as follows: Section 2 describes an efficient tree-structured codebook design, experimental results are in Section 3 and the conclusion appears in Section 4.

2. Codebook design algorithm
The technique referred to as cell merging was considered by Riskin et al. to design tree-structured VQ (TSVQ) codebooks [4]. The design procedure proceeds from a high level (of the tree) to lower levels, wherein the codevectors at the highest level are first found; the cells associated with these codevectors are then merged to form a binary tree. The cell merging technique is presented as alternative to the tree-growing technique where the codebooks are designed from the low level to higher levels and is used in [5] together with the sequential design algorithm for the design of multistage tree-structured VQ (MTVQ). Recently Chu in [6] proposed an efficient codebook search and low complexity codebook design (based on a cell-merging technique) algorithms for coding LSF parameters using MTVQ structure. We propose an efficient codebook design algorithm based on the cell-merging technique. This method consists of dividing the tree on optimal sub-trees according to a certain criterion.

Consider the single stage VQ with a (stage) codebook of size N (of resolution ) : N = log r N.

\[
Y = \{y_0, y_1, ..., y_{N-1}\}
\]

(1)

To build a (balanced) binary tree, the stage codevectors \( y_i, i = 0 \) to \( N - 1 \) are placed at the leaves’ positions of the tree corresponding to the highest level of the tree. This is an index assignment process described by

\[
e_i^{(r)} = y_{a(i,k)}; \quad i = 0 \text{ to } N - 1
\]

(2)

where the superscript notation indicates that the codevectors are placed at the level \( r \) of the binary tree. \( a(i,k) \in [0, N - 1] \) is referred to as the index assignment sequence with \( k = 0 \) to \( N! - 1 \), since with \( N \) indices there are \( N! \) permutations (sequences). Figure 1 shows an example with \( N = 8 \).

\[
\begin{align*}
&c_0^{(0)}, \\
&c_1^{(0)}, \\
&c_2^{(0)}, \\
&c_3^{(0)} \\
&c_0^{(1)}, \quad i_0^{(1)} = [0] \\
&c_1^{(1)}, \quad i_1^{(1)} = [1] \\
&c_2^{(1)}, \quad i_2^{(1)} = [10] \\
&c_3^{(1)} \\
&c_0^{(2)}, \quad i_0^{(2)} = [00] \\
&c_1^{(2)}, \quad i_1^{(2)} = [01] \\
&c_2^{(2)}, \quad i_2^{(2)} = [100] \\
&c_3^{(2)} \\
&c_0^{(3)}, \quad i_0^{(3)} = [000] \\
&c_1^{(3)}, \quad i_1^{(3)} = [001] \\
&c_2^{(3)}, \quad i_2^{(3)} = [100] \\
&c_3^{(3)} \\
&c_0^{(4)}, \quad i_0^{(4)} = [0000] \\
&c_1^{(4)}, \quad i_1^{(4)} = [0001] \\
&c_2^{(4)}, \quad i_2^{(4)} = [1000] \\
&c_3^{(4)} \\
&c_0^{(5)}, \quad i_0^{(5)} = [00000] \\
&c_1^{(5)}, \quad i_1^{(5)} = [00001] \\
&c_2^{(5)}, \quad i_2^{(5)} = [10000] \\
&c_3^{(5)} \\
&c_0^{(6)}, \quad i_0^{(6)} = [000000] \\
&c_1^{(6)}, \quad i_1^{(6)} = [000001] \\
&c_2^{(6)}, \quad i_2^{(6)} = [100000] \\
&c_3^{(6)} \\
&c_0^{(7)}, \quad i_0^{(7)} = [0000000] \\
&c_1^{(7)}, \quad i_1^{(7)} = [0000001] \\
&c_2^{(7)} \\
&c_3^{(8)}
\end{align*}
\]

Figure 1: An example of a binary tree with \( r = 3 \) (three levels). \( i_n^{(m)} \) is the index of the codeword \( c_n^{(m)} \).

For a given index assignment sequence, the cell merging process consists of merging the cells of higher-resolution to lower levels and is used in [5] together with the tree-growing technique where the codebooks are designed from the low level to higher levels and is used in [5] together with the
where \( P_{2i}^{(m+1)} \) and \( P_{2i+1}^{(m+1)} \) are the probabilities of the codevectors \( c_{2i}^{(m+1)} \) and \( c_{2i+1}^{(m+1)} \) respectively and the probability of \( c_i^{(m)} \) is equal to \( P_i^{(m)} = P_{2i}^{(m+1)} + P_{2i+1}^{(m+1)} \). \( d(x,y) \) is the distance between the vector \( x \) and the vector \( y \).

A simple way for codebook design is to exhaustively evaluate all possible index assignment sequences and the best index assignment is the one that minimizes (3). This strategy is referred to as joint full search index assignment and the number of sequences that need to be evaluated is [6]:

\[
N_1 = \prod_{i=0}^{\log_2(N/2)} \left( \frac{(N/2)^2}{2(N/2^{i+1})}! \right)^2 ; \quad N \geq 2 .
\]

For \( N \leq 8 \), the value of \( N_1 \) is small. In practice, however, the size \( N \) of the stage codebook may be quite large and the joint full search index assignment procedure becomes impractical. The problem can be resolved by dividing the stage codebook of size \( N \) into \( nN \) sub-codebooks of size \( n \leq 8 \) each. For each sub-codebook and using the joint full search, the optimal sub-tree (according to the criterion (3)) is found. Figure 2 gives an example where the stage codebook is of size \( N = 32 \). The goal here is to construct a tree of five levels: level 0 (one codevector) to level 5 (32 codevectors). From the codebook of size \( N = 32 \), four sub-codebooks of size \( n = 8 \) are found; then four optimal sub-trees are found: \( \text{subtree}_{0}^{(5,2)} \) (\( i = 0 \) to 3)

where \( \text{subtree}_{0}^{(p,q)} \) is the sub-tree constructed from level \( p \) down-to level \( q \) of the tree. Thus the sub-tree \( \text{subtree}_{0}^{(p,q)} \) has \( l = p - q \) levels (level 0 to l). Once the sub-trees \( \text{subtree}_{0}^{(5,2)} \) (\( i = 0 \) to 3) are found, the four codevectors \( c_{1,2}^{(i)} \) (\( i = 0 \) to 3 of level 2) are computed. Since at the level 2 the tree of the number of codevectors is small, the joint full search procedure may be applied to find the optimal sub-tree \( \text{subtree}_{0}^{(2,0)} \). The last designing step is the operation of index assignment, which consists of placing the codevectors on the structure of the (balanced) binary tree. The procedure described above can be generalized for a binary tree containing any number of levels.

![Figure 2: An example of the principle of designing a binary tree of five levels (level 0 to 5) by dividing the tree into sub-trees with a maximum of 3 levels each.](image)

The design procedure performance depends on the performance of the algorithm that consists of dividing a set (codebook) of \( N \) codevectors into \( nN \) sub-sets where each sub-set contains \( n \) codevectors. Let \( I_n^{(m)} \) be an index set (of index \( i \)) of \( n \) codevectors selected from \( N \) codevectors (indexed from 0 to \( N-1 \)). The number of distinct index sets (the \( n \) indices comprising the set \( I_n^{(m)} \)) must be different and the order of the indices is irrelevant) is given by \( N^2 = N^2 / n! (N-n)! \). We define the distance associated with the index set \( I_n^{(m)} \) by

\[
D[i,n] = \sum_{j \in I_n^{(m)}} P_j (c_i - y_j)^T W(y_j)(c_i - y_j) ,
\]

where \( P_j \) is the probability of the codevector \( y_j \), which is defined as the probability that input random vector \( Y \) pertains to the cell of the centroid \( y_j \) and \( W(x) \) is a diagonal weighting matrix depending on the vector \( x \). Thus \( D[i,n] \) is the sum of scaled weighted Euclidean distances between the codevectors \( y_j \) \( (j \in I_n^{(m)}) \) with respect to the vector \( c_i \). By minimizing (5), the centroid \( c_i \) is given by

\[
c_i = \left[ \sum_{j \in I_n^{(m)}} P_j W(y_j) \right]^{-1} \left[ \sum_{j \in I_n^{(m)}} P_j W(y_j)y_j \right] .
\]

The vector \( c_i \) may be interpreted as the centroid of the cell resulting from the merging of \( n \) cells having as centroids the codevectors \( y_j \) \( (j \in I_n^{(m)}) \). A sequence of \( N/n \) disjoint index sets is denoted by \( S_n^{(m)} = \{ I_n^{(m)}, i = 0 \) to \((N/n) - 1 \} \) with \( b[i,k] \in [0, N2 - 1] \) \( (k \) is the index of the sequence and \( N2 \) is the number of distinct index sets) and obviously we have:

\[
\bigcup_{i=0}^{N/2(n-1)} I_n^{(m)} = \{0,1,..,N-1\} .
\]

In the sequel \( n \) and \( N/n \) are called the dimension and the size of the sequence \( S_n^{(m)} \) respectively. To find the best sequence we minimize the distance sum

\[
D_p[k,n] = \sum_{i=0}^{(N/n)-1} D[b[i,k],n] .
\]

According to our optimality criterion (7), the best sequence \( S_n^{(m)} \) can be found by exhaustively searching through all possible sequences (the order of index sets in a sequence is irrelevant) that produce different values of the distance sum (7). This full search approach, however, is impractical because the possibilities are astronomical for moderate values of \( N \) and \( n \). The high complexity of finding the optimal sequence is due to two parameters: the number of distinct index sets \( N2 \) and the number of sequences. For example: \( N2 = 1.4297.10^{12} \) for \( N = 128 \) and \( n = 8 \). However for \( n = 2 \), \( N2 \) is reduced to 8128. We consider a suboptimal procedure that consists of finding (at the first step) the \( M_L \) low distance sequences of dimension \( n = 2 \). Then from these \( M_L \) sequences, the \( M_L \) low distance sequences corresponding to the following dimension \( n = 4 \) are found. The procedure is repeated until the desired dimension is reached and the best sequence that minimizes the overall distance (7) is found. The algorithm is given below.
Table 1. Algorithm for dividing a set of $N$ codevectors into $N/n\times n$ sub-sets of $n$ codevectors each.

- **Inputs:** the codevectors $y_j$ ($j = 0$ to $N - 1$), the probabilities $P_j$ ($j = 0$ to $N - 1$), $n1$, $N_L$ and $M_L < N_L$

1- **Initialization:** $S_0^{(1)} = [[0], [1], \ldots, [N - 1]]$, $n \leftarrow 2$ .

2- From the sequence $S_0^{(1)}$ extract a maximum of $N_L$ sequences $S_k^{(2)}$ ($k = 0$ to $N_L - 1$).

3- From the $N_L$ sequences of step 2, keep the $M_L$ distinct sequences $S_{k(j)}^{(2)}$ ($j = 0$ to $M_L - 1$) corresponding to the $M_L$ lowest values of the distance sum (7). 4- For each sequence $S_{k(j)}^{(2)}$, extract a maximum of $N_L$ sequences of dimension $2n$. Thus we have $N_L M_L$ sequences $S_{k(j)}^{(2n)}$ ($j = 0$ to $M_L 2n - 1$).

5- From the $N_L M_L$ sequences of step 4, keep the $M_L$ distinct sequences $S_{k(j)}^{(2n)}$ ($j = 0$ to $M_L - 1$) corresponding to the $M_L$ lowest values of the distance sum (7). 6- **Test:** $n \leftarrow 2n$, if $n = n1$, proceed to the next step otherwise repeat steps (4) to (5).

7- From the $M_L$ sequences of step 5, select the best sequence $S_k^{(n)}$ corresponding to the lowest value of the overall distance sum (7) at the desired dimension $nL$.

We can improve the performance of the above algorithm. Consider a sequence $S_k^{(n)}$ of $N/n$ index sets; we can exchange the indices (one index at the same time) between two index sets of the sequence $S_k^{(n)}$; then we evaluate the distance sum (7) and keep this exchange operation if (7) is minimized, otherwise the operation is cancelled. The exchange operation can be applied for all set indices and for all possible combinations of two sets contained in the same sequence $S_k^{(n)}$. For our case the index exchange operation is applied only at the desired dimension (step 7) of the algorithm for the $M_L$ sequences.

The extraction of $N_L$ sequences of dimension $2n$: $S_k^{(2n)}$ ($k = 0$ to $N_L - 1$) from a sequence of dimension $n$: $S_k^{(n)}$ can be done by generalizing the method presented in [6]: Let $S_k^{(n)} = \{I_{(n)}^{(i)}, i = 0 \text{ to } (N/n) - 1\}$. An index set of cardinality $2n$ is formed by the union operation between two index sets (of cardinality $n$) taken from the sequence $S_k^{(n)}$.

\[ \{I_{(n)}^{(i)} \cup I_{(n)}^{(j)}, i = 0 \text{ to } (N/n) - 2, j = i + 1 \text{ to } (N/n) - 1\}. \] (8)

The indices $i$ and $j$ are selected as such as the indices comprising the set of cardinality $2n$ (resulting from operation (8)) must be different. Thus we have $N3 = (N/n) [2 + (N/n) - 2)]$ (a combination of 2 indices taken from $N/n$ indices) distinct index sets of cardinality $2n$: $I_i^{(2n)}$ ($i = 0$ to $N3 - 1$). Using equation (5), the distances $D(i, 2n)$ ($i = 0$ to $N3 - 1$) associated with the sets $I_i^{(2n)}$ ($i = 0$ to $N3 - 1$) are computed. The distances are sorted in ascending order, with the associated index sets placed in the same order. For the sorted index sets and using the set with the lowest distance as the reference, eliminate all other index sets that are not disjoint with the reference. Continue the process with the next index set on the sorted list until the point where all index sets are disjoint. At the end of the process a sequence of $N/2n$ index sets $S_i^{(2n)} = \{I_i^{(2n)}, i = 0 \text{ to } (N/2n) - 1\}$ remains and forms the first sequence of dimension $2n$. We can repeatedly apply the same method to extract other sequences; we start again with the untouched sorted list, we ignore the first index set and use the second index set on the sorted list as the reference and apply the same procedure to extract another sequence. The process is repeated $N_L$ times leading to a maximum of $N_L$ different sequences $S_k^{(2n)}$ ($k = 0$ to $N_L - 1$).

Experimental results proved an improvement of the performance of the algorithm in Table 1, when including more than one sequence: $M_L > 1$ and the index exchange operation.

3. Performance evaluation

The standardized MELP coder [1],[8] uses a frame length of 180 samples, where 54 bits are used for each frame leading to a bit-rate of 2400 bps. It uses a four-stage MSVQ for LSF quantization with resolution of 7, 6, 6 and 6 bits, leading to 25 bits per frame. The training and test database are taken from the TIMIT-train and TIMIT-test database, respectively. The TIMIT sentences (clean speech) were down-sampled to 8 kHz. As in [6], we use the same codevectors as in the original standard and we apply the design codebook described in Section 2 (with $N_L = 600$ and $M_L = 40$) on a stage by stage basis. At the first stage the weighted Euclidean distance [7] is used, while for the following stages (stage 2 to 4), the Euclidean distance is used (the codevectors correspond to the residual vectors, not to the LSF vectors). The probabilities are estimated using training data of 180 914 ten-dimensional LSF vectors. Finally, we obtain a MTVQ with four stage codebooks having the same codevectors as the standard MELP quantizer (but arranged in different orders). For the voiced speech, the MELP coder depends on the computation of Fourier magnitudes (from the prediction-error signal) to capture the shape of the excitation pulse [1],[8]. The magnitudes of the Fourier transform correspond to the first 10 pitch harmonics. These 10 Fourier magnitudes are quantized with an 8-bit VQ (256 levels). We use the same codevectors as in the original standard and we apply the design codebook described in Section 2 (with $N_L = 600$ and $M_L = 40$). As in the MELP coder the 10 Fourier magnitudes are perceptually weighted with fixed weights that emphasize the low-frequency region; then the Euclidean distance is used. The probabilities are estimated using training data of 138 071 ten-dimensional vectors. Finally, we obtain a MTVQ having the same codevectors (at the highest level of the tree) as the standard MELP quantizer (but arranged in different orders).

A test data set of 87 012 LSF vectors is encoded using 25 bit MTVQ with the resulting indices decoded in an embedded manner where the number of bits ranges from 0 to 25 bits. For embedded decoding, the priority is given to the lower stages by allocating the available bits to them first (see [6]). To evaluate the quality of the quantization, the average spectral distortion (SD) and the percentage of outliers [7] for the testing data set are plotted in Figure 3. Note that there is a gradual degradation in the average SD when the number of bits is decremented, therefore confirming the LSF quantizer is bit-by-bit quantizer. A test data set of 66 743 Fourier
magnitude vectors is encoded using 8 bit TSVQ with the resulting indices decoded in an embedded manner where the number of bits ranges from 0 to 8 bits. Note that for 0 bit, the codevector at level 0 of the tree is used as default vector. To evaluate the quality of the quantization, the mean squared error (as in the MELP coder, the Fourier magnitude vector is perceptually weighted to emphasize the low-frequency region) is plotted in Figure 4. Here again we see that it changes gradually with the number used in decoding.

Fig. 3. Experimental results for a four-stage MTVQ-based LSF quantizer (25 bits). Left: Average spectral distortion (SD) as a function of the number of bits (m) used in decoding. Right: Percentage of outliers having SD in [2-4] dB and SD > 4 dB as a function of the number of bits used in decoding.

Fig. 4. Experimental results for a TSVQ-based Fourier Magnitude quantizer (8 bits). Average squared error as a function of the number of bits (m) used in decoding.

We use the ITU-T P.862 PESQ standard [9] to measure the quality of the synthetic speech produced by the FS-MELP standard coder, when the original LSF quantizer and the original Fourier magnitudes quantizer are replaced by MTVQ and TSVQ quantizers respectively. The range of the PESQ score is from -0.5 to 4.5. The speech material used for the PESQ measurement is 8.73 min of speech pronounced by 8 males and 8 females taken from the TIMIT speech data base (down-sampled to 8 Khz). Figure 5 shows the PESQ scores obtained for the modified MELP coder. Note that the 33 bits (8 for Fourier magnitudes + 25 bits for LSF’s) are decoded in an embedded manner from 0 to 33 bits. The first 25 bits (0 to 25) are used for LSF decoding and the following 8 bits (26 to 33) are used for Fourier magnitudes decoding. Thus in Fig. 5, for \( m = 0 \) to 25 no bit (0 bits) is used for the Fourier magnitude quantizer (the codevector at level 0 of the TSVQ is used by default) and for \( m = 26 \) to 33, 25 bits are used for LSF decoding. The results (Figure 5) show the PESQ scores range from 1.63 to 3.33 with an average rate of 0.05 per bit. The change in quality is minor when no bits are used for Fourier magnitudes and 19 bits are used for LSF’s. When \( m < 5 \) (only the first stage of MTVQ is used) the change in quality is more pronounced. Finally it should be noted that the tree structure allows achieving an efficient fast search algorithm at the encoding side (see reference [10]).

Fig.5. PESQ scores as a function of the number of bits (m) used in LSF and Fourier magnitudes decoding for the modified MELP coder. The speech material used in the measurement has a duration of 8.73 min.

4. Conclusion

An efficient codebook design algorithm based on the cell merging technique is proposed. Two quantizers are designed: a MTVQ for LSF’s and a TSVQ for Fourier magnitudes. To design the MTVQ and as initialization step, we use the same codevectors of the original MSVQ used by the MELP coder. To design the TSVQ and as initialization step, we use the same codevectors of the original single VQ adopted by the standard. The MELP standard is modified by replacing the original single stage VQ for Fourier magnitudes with the TSVQ and the MSVQ for LSF’s with the MTVQ. The modified coder is fine-granular bit-rate scalable (with increments of 44.4 bps) with gradual change in quality when the bit rate varies from 1288.9 bps to 2400 bps for the unvoiced speech frame and from 933.3 to 2400 bps for voiced speech frame. At 2400 bps the modified coder has exactly the same performance as the standard coder.

5. References