REDUCING THE SEARCH COMPLEXITY FOR LOW BIT RATE VECTOR QUANTIZATION BASED ON SHELLS OF GOLAY CODES

Adriana Vasilache[1], *Septimia Sârbu*[2], *Ioan Tăbuş*[2]

[1]Nokia Research Center
[2]Department of Signal Processing
Tampere University of Technology
email: adriana.vasilache@nokia.com; septimia.sarbu@tut.fi; ioan.tabus@tut.fi

ABSTRACT

We study in this paper ways to reduce the complexity required by a vector quantization scheme, which uses the shells of Golay codes for coding sub-vectors of the input vector. Such a scheme was recently shown to achieve top performance in terms of segmental SNR in the generic situation of data affected by outliers, which is relevant for the case of audio data in various sub-layers of transform based speech and audio codecs. The optimization of bit allocation and shell selection process can be simplified into a greedy algorithm with a very slight loss of performance over the exact optimal solution. We study here various algorithmic solutions, able to keep a good performance, while drastically reducing the complexity of the previously proposed full search algorithm.

1. INTRODUCTION

The interest in low bit rate quantization stems from the recent activities in ITU-T standardization of embedded scalable speech and audio codecs [1], [2] where the layered structure of the codec requires fixed rate quantization of long transform domain vectors at bit rates within the range 0.3-0.7 bits per sample (bps). Important prior work regarding the Golay codes for encoding i.i.d Gaussian sources at 1/2 bits per sample can be found in [9],[10] and [11], as well as vector quantization schemes regarding linear, nonlinear error-correcting codes and lattices, with application to audio and speech coding, in [7], [8], [10], and in the references therein.

One of the low bit rate quantization tools recently proposed in [3] is using a Golay code [6], to define a vector quantizer for long input vectors, where selected subvectors are encoded using a scalar gain, sign information, and significance information given by a Golay code word. It is convenient to define the Golay code as a codebook, organized as a union of shells or spheres, where each shell contains all code words having the same Hamming weight. The optimization process requires searching for the nearest neighbor separately within each shell of the Golay code in order to achieve a good distortion at fixed rate for the quantization scheme. However the direct search in each shell for a big number of subvectors leads to a high complexity of the overall scheme. The optimization problem can be solved exactly using linear programming, or can be (slightly) sub-optimally solved by employing a greedy search for selecting which shell of the Golay codebook to be used for a given subvector. While the approach in [3] focused in achieving as high signal to noise ratio (SNR) as possible, we deal here with the problem of lowering the complexity when using the same principles for the overall quantization scheme.

For applications where the complexity should be kept very low, an additional possibility to gradually adjust the complexity with none or very small performance degradation would be very important. We will present two sets of alternative methods for lowering the complexity. The first will address reducing the complexity of the search in one shell of the Golay code, while the second will redefine the process of bit allocation so that the search on a number of shells is restricted, depending on the energy distribution among the subvectors.

In addition to speeding up the search process, the presented approach lowers considerably the requirements of storage tables, by using the fact that the Golay codes are cyclic codes. A binary cyclic code can be seen as a union of cyclic leader classes, all defined compactly by a small number of leader vectors. The definition allows storing the codebook of the cyclic code with much less memory than it would be required by direct storage. This also allows the parametrization of the size of the codebook which makes possible defining partial search, with a various number of truncation points, and thus with a parametrizable complexity.

1.1 Low bit rate vector quantization based on Golay codes

The low bit rate quantizer described in detail in [3] is briefly reviewed next, in order to make clear the context of algorithmic extensions presented in this paper.

Long input vectors $\mathbf{x} = [x_1 \ \dots x_N]$ with $N = 280$ real valued entries need to be encoded at a given low bit rate, in the range of 0.3-0.7 bits per sample. The quantization is performed in two stages. In the first stage some entries of the vector are identified as outliers and they are quantized and transmitted together with their number, n_o, and their locations in the vector. In this paper we are not concerned with this stage due to its relatively low complexity, when compared to the second stage. In the second stage the remaining entries of $\mathbf{x}$ are grouped as $n_b = \lfloor (N - n_o)/L \rfloor$ subvectors of length L. Some of the subvectors are encoded as full zero vectors, while the rest (e.g., a generic subvector $\mathbf{z} = [z_1, \dots, z_L]^T$), are encoded using the following elements: (1) a Golay code word $\mathbf{c} \in \mathcal{G}_{23}$ for conveying the significance information ($c_i = 0$ signaling that z_i is quantized to zero); (2) the signs for all significant entries (for all i for which $c_i = 1$ a bit encodes $sign(z_i)$); and (3) an overall gain to be used with all quantized values in $\mathbf{x}$ during the reconstruction process. A mask $\{t_1, \ldots, t_{n_b}\}$ of n_b bits specifies which of the subvectors are encoded as full-zeros, and which of them are vector quantized.

We describe now more formally the encoding of a generic subvector $\mathbf{z}$: we denote j_k the sign and y_k the absolute value of the k^{th} entry so that we have $\mathbf{z} = [z_1, \ldots, z_L] = [j_1 y_1, \ldots, j_L y_L]$. The quantization of this subvector is done by using a code word $\mathbf{c} = [c_1, \ldots, c_L] \in \mathcal{G}_{23}$ to encode the significance information of the entries of z. Thus, the reconstructed output will be $\hat{\mathbf{z}} = g[j_1 c_1, \ldots, j_L c_L]^T$, where g is the gain (which is scalar quantized and encoded by 7 bits). The null code vector $\mathbf{c} \in \mathcal{G}_{23}$ will not be transmitted by a 12 bit addressing in the Golay codebook, but instead it is more efficiently specified by the single bit $t_i = 0$, which tells that the i^{th} subvector is the full-zero vector. The distortion of the reconstructed subvector is $D(\mathbf{c}, g) = \|\mathbf{z} - \hat{\mathbf{z}}\|^2 = \|[z_1, \ldots, z_L]^T - g[j_1 c_1, \ldots, j_L c_L]^T\|^2$. The bit rate r_d used for encoding one subvector will contain the $n_a = 12$ bits for addressing the Golay codebook and the additional $d_H(\mathbf{c})$ bits for the signs of significant entries of $\mathbf{z}$, so that $r_d(d_H(\mathbf{c})) = n_a + d_H(\mathbf{c})$ if $\mathbf{c} \neq \mathbf{0}$ and $r_d = 0$ when $\mathbf{c} = \mathbf{0}$. The quantity $d_H(\mathbf{c})$ represents the Hamming weight of the code word $\mathbf{c}$.

Denote now $\mathbf{z}^{[1]}, \ldots, \mathbf{z}^{[n_b]}$ the n_b subvectors of the vector $\mathbf{x}$ (after the removal of possible outliers) and use the same superscript for their associated quantities. The goal of the optimization problem with fixed rate R, stated formally as

$$\min \sum_{i=1}^{n_b} D(\mathbf{c}^{[i]}, g^*) \quad s.t. \quad \sum_{i=1}^{n_b} r_d(d^{[i]}) \leq R, \qquad (1)$$

is to find the optimal Hamming weights $d^{[1]}, \ldots, d^{[n_b]}$, the optimal codes $\mathbf{c}^{[1]}, \ldots, \mathbf{c}^{[n_b]} \in \mathcal{G}_{23}$, and the gain g^*.

1.2 A review of the greedy optimization in [3]

In order to explain the computation of distortion we introduce the scalar product, $s_{ij} = \mathbf{y}^{[i]^T} \mathbf{c}^*(\mathbf{y}^{[i]}, d_j)$, between the subvector $\mathbf{y}^{[i]}$ and its nearest neighbor code word $\mathbf{c}^*(\mathbf{y}^{[i]}, d_j)$ on the j^{th} shell of $\mathcal{G}_{23}$. A shell of $\mathcal{G}_{23}$ is the set of Golay code words having the same Hamming weight. If the optimization decision is to quantize the subvector $\mathbf{y}^{[i]}$ by its nearest neighbor code word on the j^{th} shell, we set the indicator variable $w_{ij} = 1$, while we set $w_{ij'} = 0$ for all $j' \in \{1, \ldots, n_D\}$ with $j' \neq j$, where n_D is equal to 7, the total number of shells. Thus, the solution of the optimization problem will be given in terms of the indicator variables $w_{ij} = 1$. Denoting k_b the number of non-trivially quantized subvectors, the exact solution in [3] is given as

$$\max_{w} \frac{1}{\sqrt{R - k_b n_a}} \sum_{i=1}^{n_b} \sum_{j=1}^{n_D} s_{ij} w_{ij}, \qquad (2)$$

where in each row of w there should be only one nonzero entry. The possible values of k_b are just a few and for each k_b we must solve a linear programming problem.

A simplified greedy algorithm was shown to provide solutions very close to the optimal ones. We summarize the algorithm for the Golay code $\mathcal{G}_{23}$, where the admissible rates $r_d(d_H(\mathbf{c})) = n_a + d_H(\mathbf{c})$ belong to the set $R_d = \{19, 20, 23, 24, 27, 28, 35\}$.

```
Algorithm 1
1. Generate iteratively all possible combinations (tuples) $d_{i_1} \geq d_{i_2} \geq \ldots \geq d_{i_{k_b}}$ with $r_d(d_{i_j}) \in R_d$ such that $\sum_{j=1}^{k_b} r_d(d_{i_j}) = R$.
2. For each such tuple, say $V = (d_{i_1}, d_{i_2} \ldots d_{i_{k_b}})$, we first check which are the optimizing arguments $\ell^*, i_{j*} = \arg\max_{\ell, i_j} \{s_{\ell i_j} : \ell = 1, \ldots, n_b, \quad i_j \in \{i_1, \ldots, i_{k_b}\}\}$ and we allocate to the subvector $\ell^*$ the code word located on the ${i_{j*}}^{\text{th}}$ shell.
3. We remove now the subvector $\ell^*$ from the list of subvectors and $d_{i_{j*}}$ from the tuple $V$ and continue the process.
```

Generating the list in Step 1 is a simply recursive routine. As an example, if the available bitrate is $R = 80$ bits and the binary code is $\mathcal{G}_{23}$, with $R_d = \{19, 20, 23, 24, 27, 28, 35\}$, there are 3 such tuples, namely two for $k_b = 4$: $V_1 = (8, 8, 8, 8)$, $V_2 = (11, 7, 7, 7)$, and one for $k_b = 3$, $V_3 = (16, 16, 12)$.

2. FAST SEARCH USING CYCLIC LEADERS IN A SHELL OF A GOLAY CODE

We will exemplify the definition and use of the cyclic leaders in the case of the Golay codes, but the principle of the fast search method presented here applies to any binary cyclic code. The binary Golay code of dimension 23 can be seen as a union of several 23-dimensional spheres, which we call shells, $\mathcal{G}_{23} = \bigcup_{i \in \mathcal{D}} \mathcal{S}_i$ with $\mathcal{D} = \{0, 7, 8, 11, 12, 15, 16, 23\}$. Each shell or sphere has radius $i \in \mathcal{D}$ called weight of the shell (or Hamming weight), representing the number of ones that are in each code word from that shell. The shells having weights 0 and 23 have only one code word each, corresponding to the all zero and all ones vectors, respectively and they are considered trivial (they require negligible cost of search, when compared to the rest of 4094 code words). The numbers of code words on the rest of the shells are $N_7 = N_{16} = 253, N_8 = N_{15} = 506, N_{11} = N_{12} = 1288$. Each of the pairs of shells (7; 16), (8; 15), and (11; 12) possesses the following complementarity property: in each pair of shells, the vector obtained by complementing the binary entries of any code word in the first shell will be a valid code word belonging to the second shell, e.g., $00010010101101001011011 \in \mathcal{S}_{11}$ will be transformed by complementing the bits into $11101101010010110100100 \in \mathcal{S}_{12}$.

By the definition of the cyclic code [6], for any given code word from one shell of the 23-dimensional Golay code, 22 other code words from the same shell can be obtained through cyclic shifts. Therefore, for each shell having N_i code words, one can choose $N_i/23$ code words, which can generate the entire shell by cyclic shifts. These vectors are dubbed here *cyclic leader vectors* and the set of code words obtained through all the cyclic shifts of one cyclic leader vector is referred to as cyclic leader class of that vector. Each shell of a binary Golay code can be defined as a union of cyclic leader classes. As a consequence, the entire Golay code can be represented as a union of cyclic leader classes.

The Golay code is not unique and there are several generator matrices for it. Thus, the set of cyclic leader vectors used to define a Golay codebook is not unique. Moreover, even for a given generator matrix, every code vector from a cyclic leader class can be considered a cyclic leader vector. The principle we use in the following can be applied no matter the particular implementation of the codebook and cyclic vector selection.

We are storing for each shell of the Golay code a bi-dimensional array of integers, where the integers on each line of the array represent the position of the units in one of the cyclic leader vector. There are only three arrays defining the six non-trivial shells, because we use the complementarity property, such that the array which indicates the positions of ones for the shell of weight 7 will also indicate the positions of zeros for the shell of weight 16 (the same applies for the other two arrays, for the pair 8-15 and for the pair 11-12).

The nearest neighbor search in a shell can be performed as exemplified in the following. For the search in a given shell, the distortion minimization can be replaced by the maximization of the scalar product between the input vector and the code words, given the fact that within the same shell all code vectors have the same Hamming weight. The search is done simultaneously in the two shells, based on the property of complementarity, such that the scalar product for the complementary shell is obtained by subtracting the scalar product computed for the first shell of the pair, from the sum of the components of the input vector. The two search procedures can be separated if the nearest neighbor is needed only for one of the shells.

There are two main loops in the search procedure, one over all the cyclic leader vectors defining the shell and the second one over all the cyclic shifts for a given leader. The result of the search procedure is represented by the optimal leader vector indexes, for each of the two shells, and the indexes of the optimal cyclic shifts for the corresponding leader vectors. The shells with the highest complexity are those of weight 11 and 12 and the complexity of search in weighted million operations per second (WMOPS) for these shells is relatively the same as for a fast method based on the hexacode [4] devised along the lines of [5].

The method presented here allows further reducing the complexity by adjusting the number of cyclic leaders or the number of cyclic shifts on which the search is performed. In the context of the quantization tool presented in [3], the performance drop due to the reduction of the codebook can be compensated by the fact that there are actually less bits needed to encode a Golay code word and potentially more sub vectors can be represented by Golay code words. The overall complexity of this method compared to that of a direct search on each shell is more than halved.

In view of the fact that not all the cyclic shifts are considered in the search, the arrays defining the codebook can be optimized such that the ones in the code words are evenly covering the 23-dimensional space.

3. RESTRICTING THE NUMBER OF SHELLS BASED ON SUBVECTOR NORMS

Prior to the evaluations for each tuple V in step 2 of Algorithm 1, the scalar products $s_{ij} = {y^{[i]}}^T c^*(y^{[i]}, d_j)$ should be estimated for each subvector and each shell of the Golay code, procedure which demands a relatively large computational effort. There are several heuristic observations which help to reduce the complexity by restricting the number of scalar product calculations.

First of all, the small number of available bits practically restricts the maximum number of subvectors which are coded by Golay code words to a value lower than n_b. In the optimization process, the most likely subvectors to be quantized by Golay code words are those having higher energy, therefore we can reduce the number of subvectors for which the scalar product is evaluated.

Secondly, it is less likely that a subvector with high energy will be coded by a Golay code word from a lower Hamming weight shell, and similarly there is smaller probability that a subvector with low energy will be quantized by a Golay code word from a higher Hamming weight shell.

In the nearest neighbor search algorithm on Golay code shells, the shells with the highest complexity requirements are the shells of Hamming weights 11 and 12. Therefore we try to find ways in which we can reduce the usage of shells 11 and 12.

With the help of these heuristics we can devise several lower complexity variants of the quantization method, that will be detailed in the experimental section. The resulting methods will have significantly lower complexity, but they will entail also a reduction in compression performance.

In addition, there are two bit-exact modifications that help reduce the complexity, but by a lower amount. They relate to the generation of the tuples V, which we decide to pre-store for the most frequent number of available bits (which depends on the most frequent number of outliers). Apart of these intra-frame savings we can achieve inter-frame savings by checking from a long input vector to the next if the number of outliers is the same, allowing to use the same array of tuples.

4. EXPERIMENTAL RESULTS

Throughout the experimental section we consider two data sets. The first contains 14167 long vectors (frames) with 280 components each derived from a wide range of speech and music material. The entries in the vectors are differential MDCT coefficients in a layer of an embedded scalable experimental audio codec. The second set is similarly derived from the speech and music files from Table 3.

The performance is evaluated in terms of distortion by segmental SNR (average of SNR over all the frames) and complexity evaluated in WMOPS.

The original method [3] had, on the first test set, a segmental SNR of 2.59 dB for a bit rate of 144 bits per frame, whereas the Voronoi extension to the RE_8 lattice [7] benchmark method had a segmental SNR of 2.36. The experimental results of this paper are compared to those of the original method and further benchmark results can be found in [3].

The complexity of the method implemented prior to any proposed complexity reduction techniques has been evaluated to 24 WMOPS.

By using the proposed limitation at 6 for the number of blocks to be coded by Golay code words, the complexity has been brought down to 12 WMOPS, and the performance has remained unchanged.

We consider next the complexity reductions by restricting the number of shells. For the case of 144 available bits (without the 7 bits needed for the quantization of the overall gain), there are at most 6 subvectors from the 280-dimensional vector that are coded by Golay code words. The considered variants are described next. For all the variants the subvectors are decreasingly sorted with respect to energy.

Variant A1: assuming that the first 3 out of the 6 subvectors have significantly higher energy, we compute the scalar product for subvectors 1-3 only for shells 8, 11, 12, 15. For the subvectors 4-6 which have smaller contribution to the overall distortion, we decide to skip the very expensive search on the shells 11 and 12.

Variant A2: we compute the scalar product for subvectors 1-4 only for the shells 8, 11, 12, 15. For the subvectors 5-6 we skip the very expensive search on the shells 11 and 12.

Variant	Nr. of subvect.	Shells (7,8,11,12,15,16)	SNR	Cplx. (WMOPS)
A1.	1-3	(8,11,12,15)	2.54	7.3
	4-6	(7,8,15,16)		
A2.	1-4	(8,11,12,15)	2.55	8.3
	5-6	(7,8,15,16)		
A3.	1-3	(7,8,11,12,15,16)	2.55	7.9
	4-6	(7,8,15,16)		
A4.	1-4	(7,8,11,12,15,16)	2.56	9.2
	5-6	(7,8,15,16)		
A5.	1-3	(11,12)	2.51	5.9
	4-6	(7,8,15,16)		

Table 1: Segmental SNR and complexity evaluation for the methods based on the reduction of the number of shells.

For the remaining variants the restrictions on shells are listed in Table 1. All variants include the previously presented methods for bit exact complexity reduction.

The complexity values, in terms of WMOPS, and the corresponding segmental SNR values are presented for each of the five variants in Table 1. The difference between variant A1 and A2 is that an additional subvector is encoded using shells 8, 11, 12, 15, which leads to an increase of 1 WMOPS in complexity and an insignificant increase in performance, due to the fact that the fourth subvector has less energy than the first three subvectors and, thus, does not bring a significant contribution to the overall performance. Therefore, the usage of shells 11 and 12 for subvectors with lower energy is unjustified if we want to drastically reduce the complexity of the schemes. The same principle applies to the next set of variants, A3 and A4.

Since the variant A1 has reasonable complexity reduction and just a slight performance degradation, we select it for the rest of the experiments, performed with the second data set.

We proceed to testing the variants provided by the reduction of the number of cyclic shifts and the reduction of the number of leaders considered for each shell. In these experiments the total number of available bits is 153 (without the 7 bits for the quantization of the overall gain).

In the first approach (A11) we perform only half of the cyclic shifts of each of the leader vectors and, as expected, the complexity is halved with respect to variant A1, but the SNR value is diminished. It should be noted that when only half of the cyclic shifts are considered for each leader vector, 11 bits are enough to encode the resulting Golay code words. Results in terms of compression performance and complexity are presented in Table 3. The compression and complexity values are presented also for the encoding based on the Voronoi extension (RE_8).

Shell	Number of cyclic shifts	Number of leader vectors
7, 16	23	11
8, 15	20	22
11, 12	1	56
23	1	1

Table 2: Number of cyclic shifts for the leader vectors from each shell.

The second approach (A12) considers a more drastic reduction of the number of cyclic shifts presented for the leader vectors in each shell in Table 2.

The resulting Golay code based codebook can be divided in two parts. The first part contains the code vectors generated by the leaders in shells of Hamming weight 7, 16, and 23 (i.e. $11 \cdot 23 + 11 \cdot 23 + 1 = 507$ code vectors) and the second part contains the code vectors generated by the leader vectors from shells 8, 11, 12, and 15 (i.e. $22 \cdot 20 + 56 \cdot 1 + 56 \cdot 1 + 22 \cdot 20 = 992$ code vectors). The first set can be addressed using 9 bits and the second one using 10 bits; one additional bit is used per subvector to specify the set.

A third variant (A13) consists in halving the number of leader vectors for each shell, thus reducing to 11 the number of bits used to represent the Golay code words obtained by all cyclic shifts of the selected leader vectors. The results for variants A12 and A13 are presented in Table 3.

It can be observed that both variants A11 and A13 reduce the complexity to half with respect to A1, but there is also a significant degradation in SNR.

With the exception of three files that are characterized by very energetic outliers, the Golay based methods have better compression performance. The encoding complexity is higher than for the Voronoi extension based method. However, the decoding for the proposed methods based on Golay codes is faster than the decoding for the Voronoi extension based method (0.1 WMOPS compared to 0.28 WMOPS).

File	Description	A1		A11		A12		A13		RE_8 [7]	
		SNR (dB)	Cx.	SNR (dB)	Cx.	SNR (dB)	Cx.	SNR (dB)	Cx.	SNR (dB)	Cx.
es01	Vocal (S. Vega)	**2.86**	7.4	2.83	4.4	2.85	*4.1*	2.82	4.4	2.82	1.3
es02	German male speech	**2.84**	7.4	2.80	4.4	**2.84**	*4.1*	2.80	4.4	2.63	1.3
es03	English female speech	**3.03**	7.4	2.99	4.4	3.02	*4.1*	2.98	4.4	2.85	1.3
sc01	Trumpet solo and orch.	**3.88**	7.4	3.82	4.4	**3.88**	*4.1*	3.82	4.4	3.82	1.3
sc02	Classical orch. music	3.12	7.4	3.08	4.5	**3.12**	*4.1*	3.08	4.4	2.97	1.3
sc03	Contemp. pop music	2.69	7.5	2.66	4.5	**2.70**	*4.1*	2.66	4.4	2.51	1.3
si01	Harpsichord	**3.00**	7.4	2.97	4.4	**3.00**	*4.1*	2.96	4.4	2.93	1.3
si02	Castanets	3.00	7.4	2.97	4.4	2.99	*4.1*	2.95	4.4	**3.18**	1.3
si03	Pitch pipe	3.40	7.4	3.36	4.4	3.40	*4.1*	3.35	4.4	**3.42**	1.3
sm01	Bagpipes	3.72	7.4	3.70	4.4	3.72	*4.1*	3.67	4.4	**3.84**	1.3
sm02	Glockenspiel	4.61	7.2	4.51	4.2	4.52	*3.9*	4.44	4.2	**6.10**	1.2
sm03	Plucked strings	3.08	7.5	3.04	4.5	**3.09**	*4.1*	3.04	4.4	2.87	1.3

Table 3: Segmental SNR and complexity evaluation for the methods based on the shells number reduction (A1), based on the cyclic shifts number reduction (A11, A12), and based on the leader vector number reduction (A13).

The best performer is the method A12, where the number of cyclic shifts is drastically reduced for the shells 11 and 12 (which have the highest complexity requirements). The reduction in complexity is substantial, and the segmental SNR is even increased due to different bit consumption by the Golay code vectors.

5. CONCLUSION

In this work we have presented several approaches of reducing the complexity of a low bit rate quantization scheme based on shells of Golay codes. The most time consuming procedures, namely the search on Golay shells and the process of bit allocation are made faster by using properties of the Golay codes represented as union of cyclic leader classes and by restricting the bit allocation search process based on the energy distribution among subvectors. The complexity is 5 times reduced, while preserving the very good compression efficiency of the original method.

REFERENCES

[1] T. Vaillancourt, M. Jelínek, A. E. Ertan, J. Stachurski, A. Rämö, L. Laaksonen, J. Gibbs, U. Mittal, S. Bruhn, V. Grancharov, M. Oshikiri, H. Ehara, D. Zhan, F. Ma, D. Virette, S. Ragot “ITU-T EV-VBR: A robust 8-32 kbit/s scalable coder for error prone telecommunications channels” in *Proceedings of EUSIPCO 2008*, Lausanne, Switzerland, August, 25-29, 2008.

[2] S. Ragot, B. Kövesi, R. Trilling, D. Virette, N. Duc, D. Massaloux, S. Proust, B. Geiser, M. Gartner, S. Schandl, H. Taddei, Y. Gao, E. Shlomot, H. Ehara, K. Yoshida, T. Vaillancourt, R. Salami, M. S. Lee, and D. Y. Kim “ITU-T G.729.1: An 8-32 kbits/s scalable coder interoperable with G.729 for wideband telephony and voice over IP” in *Proceedings of ICASSP 2007*, Honolulu, Hawaii, USA, April, 15-20, 2007.

[3] I. Tabus and A. Vasilache “Low bit rate vector quantization of outlier contaminated data based on shells of Golay codes” in *Proceedings of DCC 2009*, pp. 302-311, Snowbird, USA, March 15-18, 2009.

[4] V. Pless “Decoding the Golay codes”, *IEEE Transactions on Information Theory,* vol. 32, no. 4, pp. 561-567, 1986.

[5] A. Vardy and Y. Be’ery “More efficient soft decoding of the Golay codes”, *IEEE Transactions on Information Theory,* vol. 37, no. 3, pp. 667-672, 1991.

[6] F.J. MacWilliams and N.J.A. Sloane “The theory of error-correcting codes” Amsterdam, North-Holland, 762 p., 1977.

[7] S. Ragot, B. Bessette and R. Lefebvre “Low-complexity multi-rate lattice vector quantization with application to wideband TCX speech coding at 32 kbit/s” in *Proceedings of ICASSP 2004*, vol. 1, Montreal, Canada, May 17-21, 2004.

[8] J.H. Conway and N.J.A. Sloane “Soft decoding techniques for codes and lattices, including the Golay code and the Leech lattice”in *IEEE Transactions on Information Theory,* vol. IT-32, no. 1, pp. 41-50, January 1986.

[9] S. Ragot, J.-P. Adoul, and R. Lefebvre “Hexacode-based quantization of the Gaussian source at 1/2 bit per sample”, *IEEE Transactions on Communications,* vol. 49, no. 12, pp. 2056-2058, December 2001.

[10] J.-P. Adoul and C. Lamblin “A comparison of some algebraic structures for CELP coding of speech”in *Proceedings of ICASSP 1987*, vol. 12, pp. 1953-1956, Dallas, Texas, USA, April 06-09, 1987.

[11] Z. Ben-Néticha, P.Mabileau and J.-P. Adoul “The ”Streched”-Golay and other codes for high-SNR finite-delay quantization of the Gaussian source at 1/2 bit per sample”, *IEEE Transactions on Communications,* vol. 38, no. 12, pp. 2089-2093, December 1990.