Quantization of LSF by Lattice Shape-Gain Vector Quantizer

Jianqin Chen, Ioan Tabus and Jaakko Astola
Signal Processing Lab., Tampere University of Technology, Finland

ABSTRACT

This paper presents a new lattice vector quantizer, namely lattice based shape-gain vector quantizer (LSGVQ), for line spectral frequencies (LSF) quantization. LSGVQ is developed in an attempt to reduce computation time for codebook search and memory requirement for saving codewords. Optimal design method and optimal encoding algorithm are provided for LSGVQ based on optimization of the squared error distortion. Experiments show that LSGVQ for LSF quantization can achieve spectral distortion less than 1 dB at 20 bits/frame and transparent quality at 26 bits/frame.

1. INTRODUCTION

LPC (Linear Predictive Coding) has been successfully used in speech coding to adaptively represent the spectral envelope of speech. LPC parameters are the coefficients of an p^{th} order all pole synthesis filter, usually p=10, and are transformed into other forms of parametric representations for efficient and stable coding, among which LSF (Line Spectral Frequencies) representation is mostly used in various speech codes due to its distinct features of stability and efficient coding.

Many schemes have been studied to explore the way for efficient quantization of LSF parameters while trying to reduce the codebook search time and memory utilization for saving codewords. Paliwal and Atal [1] used the split quantizatin method (SVQ) to encode LSF and achieved transparent quality quantization at about 26 bits per frame for the Euclidean distance measure and at about 24 bits per frame for the weighted Euclidean distance measure. The transparent quantization defined in [1] satisfies three conditions, i.e., 1) The average distortion is about 1 dB; 2) There is no outlier frame having SD larger than 4 dB; and 3) The number of outlier frames with SD in the range 2-4% dB is less than 2%. Multistage vector quantization (MSVQ) was adopted in [2] to quantize LSF and the same performance virtually identical to the 24 bit split VQ was obtained at 22 bits/frame. Using two stage VQ-Lattice VQ (VQ-LVQ), Pan and Fischer[3] compared their method with SVQ and MSVQ, and provided better performance than [1] and [2]; they got the transparent quantization at rates ranging in 24-25 bits/frame. All these algorithms still need significant memory to save codebook and the computation time to search the nearest neighbor for any input vector is still considerable.

To further reduce the memory requirement for the storage of codebooks and the computation complexity for searching codeword in vector quantization of LSF, lattice vector quantization is chosen as a very promising alternative. Several publications have addressed the lattice VQ for LSF [3,4,5,6].

In this paper, we consider the LSF quantization using lattice based VQ. We develop our quantizer along the line of the shape-gain vector quantizer [7,8,9] by replacing a normal shape quantizer with a lattice based VQ. Our lattice based shape quantizer is used to quantize the shape of an input vector and, at the same time, has low computation for codebook search and no memory requirement for storage of codebook. The optimal design of the quantizer under the mean squared error distortion measure is presented, along with the optimal encoding algorithm. TIMIT speech database is used for training and testing the proposed quantization scheme and experiments show that the quantizer can reach SD less than 1 dB at 20 bits/frame and transparent quality at 26 bits/frame.

2. LATTICE BASED SHAPE-GAIN VECTOR QUANTIZER

Lattice based shape-gain vector quantizer (LSGVQ) is developed based on the general concept of shape-Gain VQ [7,8,9], and the difference is that the shape VQ in LSGVQ is realized by a lattice VQ called lattice based shape vector quantization (LSVQ).

LSVQ is a lattice based VQ with all codewords having unit gain. A lattice with several spherically truncated shells can be selected to form an LSVQ. We express the LSVQ as

$$LSVQ = \left\{ QX_0 = N(QX) \mid QX \in \Lambda, n(QX)^2 \in \{m_1, m_2, ..., m_n\} \right\} \quad (1)$$

where QX_0 is the codeword of the LSVQ with unit gain , $N(QX)$ is the normalization of a lattice vector QX , $n(QX)$ is the norm of QX , Λ denotes a lattice, and m_i is the squared radius of a sphere which is used for truncation of a lattice. The combination of several different lattice shells which are scaled by their own radii generates a unit lattice VQ. With our special application to LSF, the lattice Λ is selected as

$$D_{10}^{+} = D_{10} \cup \left(D_{10} + \left\{ \frac{1}{2}^{10} \right\} \right) \quad (2)$$

where

$$D_{10} = \{(x_1, x_2, ..., x_{10}) \in Z^{10} \mid \sum x_i = even\} \qquad (3)$$

The fast search algorithm of the LSVQ is based on the original fast quantizing algorithms [10]. The derived search algorithm needs to find all possibly existed lattice points on different shells as a possible nearest neighbor (NN) and choose the one which has the optimal distortion. The search direction is always arranged from the outmost shell to the innermost one and the search step is taken equal to the distance between the adjacent shells. Generally, the outmost shell should be selected as the one with lattice points on it, because the outmost shell without any lattice point on it will lead to a search at most as good as in the case where the outmost shell has lattice points. Since an input vector to the shape quantizer always has unit gain, we scale the input by the radii of shells of the LSVQ and then use the original search algorithm to get the nearest neighbour (NN) of the input. The fast search algorithm in the LSVQ is described in detail as follows

Step0. Given the input vector X_0 with unit gain and all squared radii of the shells in an LSVQ, i.e., $m_i (i = n, n-1, ..., 2, 1)$ where $m_i > m_{i-1}$.

Step1. Multiply X_0 by the squared root of m_n to get $X = X_0 \cdot \sqrt{m_n}$.

Step2. Find the nearest neighbor QX in the lattice for the vector X using the algorithm of Conway and Sloane [10].

Step3. If $n(QX)^2 \in m_i (i = n, n-1, ..., 2, 1)$, then go to Step 4; Otherwise $n = n-1$ and go to Step 1.

Step4. Output $QX_0 = N(QX)$ as the shape VQ result

In order to get a better distortion performance, more shells can be appended to the set $m_i (i = n, n-1, ..., 2, 1)$, even shells where there is no lattice point at all, especially those shells near the outmost lattice shell. If there exist some input vectors for which no nearest neighbor can be found, more lattice shells with lattice points should be added to the LSVQ. Consequently, the associated bit rate increases. Moreover, the truncated shells of an LSVQ should be chosen to let the distance between the largest and smallest shells be larger than the covering radius of the employed lattice. Finally, in most cases, an NN can be found by search in the first few spheres for a given LSVQ.

The presented LSVQ can be used to build the overall LSGVQ. Suppose $\hat{S}_n$ is an LSVQ and $\hat{g}_n$ is a scalar quantizer with a few codewords. A sub-LSGVQ is defined as

$$C_n = \{\hat{S}_n; \hat{g}_n\} = \{\hat{S}_n(i), i = 1,2,..., N_{ns}; \hat{g}_n(j), j = 1,2,..., N_{ng}\} \qquad (4)$$

where N_{ns} is the number of lattice points in each $\hat{S}_n$ and N_{ng} is the number of codewords in each $\hat{g}_n$.

An LSGVQ consists of several sub-LSGVQ's, i.e.,

$$C = \{C_n; n = 1,2,..., N\} \qquad (5)$$

One example of an LSGVQ is

$\hat{S}_1 = lattice\ E_8$ with shells $m = \{4,3,2\}$, N_{1s} =2400;
$\hat{g}_1 = \{4.25\}$, N_{1g} =1
$\hat{S}_2 = lattice\ E_8$ with shells $m = \{6,5,4,2\}$, N_{2s} =9120;
$\hat{g}_2 = \{1.66\}$, N_{2g} =1
$\hat{S}_3 = lattice\ E_8$ with shells $m = \{8,7,6,4,2\}$, N_{3s} =26640;
$\hat{g}_3 = \{2.44, 3.32\}$, N_{3g} =2

Given an input X, the quantized output $\hat{X}$ is the product of $\hat{g}_n(j)$ and $\hat{S}_n(i)$, resulting in the squared error distortion

$$d(X, \hat{X}) = \left\| X - \hat{X} \right\|^2 = \left\| X - \hat{g}_n(j) \cdot \hat{S}_n(i) \right\|^2 \qquad (6)$$

In order to obtain an optimal LSGVQ, several design steps should be iterated as described in [9] for the shape-gain VQ design. Because the fixed LSVQ's have been selected in advance in the design process, only the gain quantizers associated with each LSVQ need to be optimized according to (6) which results in the following updating formula

$$\hat{g}_n(j) = \frac{1}{|C_n(j)|} \sum_{X \in C_n(j)} X^T \cdot \hat{S}_n(i) \qquad (7)$$

Where $C_n(j)$ is the partition region of $\hat{g}_n(j)$ and $\hat{S}_n$, and $|C_n(j)|$ is the number of vector X falling into $C_n(j)$. The corresponding sequential updating way is

$$\hat{g}_n(j) = \hat{g}_n(j) + \alpha\left(X^T \cdot \hat{S}_n(i) - \hat{g}_n(j)\right) \qquad (8)$$

Where α is a small constant, say 0.006. Thus the design algorithm for LSGVQ is proposed as follows

Step0. Choose a set of LSVQ's, say $\hat{S}_n$ $(n = 1,2,..., N)$, which meet the bit rate requirement and initialize the corresponding gain quantizers, say $\hat{g}_n(j)$ $(j = 1,2,..., N_{ng})$ $(n = 1,2,..., N)$, by applying the Lloyd algorithm to gains, $n(X)$, of input sources.

Step1. Encode an input X as described later to determine index (n, i, j); Compute the optimal codewords using (8); Repeat Step1 for all input

vectors.

Step2. If $\{\hat{S}_n, \hat{g}_n\}, n = 1,2,..., N$ do not change, Stop. Else go to Step1.

The optimal encoding scheme for LSGVQ is based on the above optimal processes, which is depicted in Fig. 1.

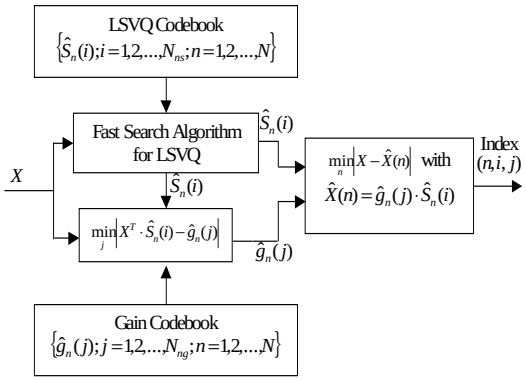

Fig. 1 An Optimal Encoding for LSGVQ

The Optimal Encoding Algorithm for LSGVQ:

S-1 For n =1 to N

- Find the shape code $\hat{S}_n(i)$ in $\hat{S}_n$ for a given input vector X by using the Fast Algorithm of LSVQ;
- compute $X^T \cdot \hat{S}_n(i)$;
- find a gain code $\hat{g}_n(j)$ for each $\hat{S}_n$ by minimizing $\left|X^T \cdot \hat{S}_n(i) - \hat{g}_n(j)\right|$;
- compute the reconstruction $\hat{X}(n) = \hat{g}_n(j) \cdot \hat{S}_n(i)$ and its distortion $d(n) = \left\|X - \hat{X}(n)\right\|^2$.

End

S-2 Find the final reconstruction as $\hat{X} = \hat{g}_n(j) \cdot \hat{S}_n(i)$ by minimization of $d(n)$ $(n = 1,2,..., N)$. The index (n, i, j) will be the encoding result.

3. QUANTIZATION of LSF by LSGVQ

LSF parameters are an equivalent representation of LPC and are encoded for efficient transmission. The encoding performance is measured by average spectral distortion (SD) which is defined as

$$SD = \left\{ \frac{1}{N_f} \sum_{n=1}^{N_f} \left(\frac{1}{\Theta} \int_0^{\Theta} \left[10\log_{10} \left| A_n(e^{j\omega}) \right|^2 - 10\log_{10} \left| \hat{A}_n(e^{j\omega}) \right|^2 \right]^2 d\omega \right) \right\}^{1/2} \quad (9)$$

where N_f is the total number of frames, $A_n(e^{j\omega})$ and $\hat{A}_n(e^{j\omega})$ are the spectra of the nth speech frame without and with quantization, respectively, and Θ is $3\pi/4$ for the 3kHz frequency band. $A_n(e^{j\omega})$ and $\hat{A}_n(e^{j\omega})$ are computed by using 256 point FFT.

Using LGSVQ to quantize LSF parameters, the following special considerations need to be taken. First, since the values of LSF parameters are distributed between $(0, \pi)$, their average is subtracted from each set of LSF so that the shape of the difference will be uniformly distributed around the unit sphere. Thus quantization by LSGVQ will be more efficient. Secondly, when using LSGVQ to quantize LSF, the outliers most likely occur at the space where gain is bigger, so LSVQ with a larger bit rate should be allocated to a big gain in order to reduce the number of outliers. Finally, since the norm of the difference between LSF and their average varies in a very small range, the total number of codewords for all gain quantizers (each associated with one LSVQ) is usually chosen between 6 to 10, depending on the distribution of the bit rates of the associated LSVQ's.

4. EXPERIMENTS

We follow the same steps in CS-ACELP [11] to obtain LSF parameters for every 10 ms speech analysis frame, including highpass filter preprocessing, windowing, and computing LPC parameters by Levinson algorithm. The only difference is in the way to deal with the bandwidth expansion. In our experiments, the 10 Hz bandwidth expansion is used by multiplying each LPC parameter a_i $(i = 1,2,...,10)$ by 0.996^i [3].

Speech data for experiments are from TIMIT database, and 5,526 frames in the trainining database are used for training LSGVQ and 124,201 frames in the test database for testing our quantization method. Several cases with different bit rates used for LSF quantization have been studied, and in each case the optimal LSGVQ is obtained by the optimal design method developed above. In the design process, LSVQ's with a big bit rate are selected with priority on the condition that the total bits requirement is satisfied and total number of codewords in gain quantizer is in between 6 and 10. Taking the bit rate 18 as an example, the corresponding LSGVQ is shown below, with a total of six gain codewords, the smallest of which is associated with a small bit rate LSVQ and other five associated with a big bit rate LSVQ. The bit rate of the LSGVQ is 17.9428.

$\hat{S}_1 = \mathrm{D}_{10}^{+}$ with shells $m = \{4.5,4,3,2.5,2\}$, N_{1S} =9192

$\hat{g}_1 = \{0.0929\}$, N_{1g} =1

$\hat{S}_2 = \mathrm{D}_{10}^{+}$ with shells $m = \{6.5,6,5,4.5,4,2.5,2\}$, N_{2S} =48552;

$\hat{g}_2 = \{0.1413, 0.2368, 0.3438, 0.4942, 0.6823\}$, N_{2g} =5

The experiment results are shown in Table 1 which are obtained by the optimal encoding scheme in Fig 1.

Talbe 1 Average Spectral Distortion

bits	SD (dB)	>1 (%)	>2 (%)	>4 (%)
18	1.1110	35.69	9.65	1.27
19	1.0543	33.83	8.28	0.74
20	0.9788	29.45	7.26	0.68
21	0.9085	25.70	5.89	0.35
22	0.8627	22.50	4.95	0.21
23	0.7953	19.44	4.22	0.15
24	0.7569	17.54	3.63	0.07
25	0.7137	16.12	2.99	0.02
26	0.6683	13.82	2.37	0.01

As shown in Table1, our LSF quantization scheme can achieve the SD less than 1 dB at 20 bits/frame and transparent quality at 26 bits/frame, while retaining the appealing features of lattice vector quantization, i.e., fast search algorithm and no memory required to save the codebook. The LSGVQ for 20 bits/frame is given below:

$\hat{S}_1 = \mathrm{D}_{10}^{+}$, $m = \{6, 5.5, 5, 4.5, 4, 2.5, 2\}$, N_{1s} =48552

$\hat{g}_1 = \{0.0739, 0.0963, 0.1167\}$, N_{1g} =3

$\hat{S}_2 = \mathrm{D}_{10}^{+}$, $m = \{8.5, 8, 7, 6.5, 6, 4.5, 4, 2.5, 2\}$, N_{2s} =167132;

$\hat{g}_2 = \{0.1437,\ 0.2292, 0.3247,\ 0.4517\}$, N_{2g} =4

$\hat{S}_3 = \mathrm{D}_{10}^{+}$, $m = \{10, 9, 8.5, 8, 6.5, 6, 4.5, 4, 2.5, 2\}$, N_{3s} =296196;

$\hat{g}_3 = \{0.6424\}$, N_{3g} =1

and for 26 bits/frame the LSGVQ is specified by

$\hat{S}_1 = \mathrm{D}_{10}^{+}$,

$m = \{18.5, 18, 17, 16.5, 16, 14.5, 14, 12.5, 12, 10.5, 10, 8.5\}$, N_{1s} =6585488

$\hat{g}_1 = \{0.1089, 0.1542, 0.2218, 0.2918, 0.3718, 0.4687\}$, N_{1g} =6

$\hat{S}_2 = \mathrm{D}_{10}^{+}$,

$m = \{20, 19, 18.5, 18, 16.5, 16, 14.5, 14, 12.5, 12, 10.5, 10\}$, N_{2s} =8569272

$\hat{g}_2 = \{0.5884, 0.7248, 0.9068\}$, N_{2g} =3

For each bits/frame, several different LSVQ's have been selected and, accordingly the associated LSGVQ's are designed using the optimal design algorithm. The computation results with different LSGVQ's for the same bits/frame have shown that similar SD and outlier percenatge are obtained by many different settings in the LSGVQ.

5. CONCLUSION

We have developed a lattice based shape-gain vector quantization algorithm in order to reduce the codebook searching time and the storage memory for the codebooks. The optimal design method and optimal encoding algorithm for LSGVQ are presented, which can be successfully used for applications like LSF quantization. The experiments with TIMIT speech database for LSF quantization show that LSGVQ can obtain the SD less than 1 dB at 20 bits/frame and transparent quality at 26 bits/frame while retaining the useful features of lattice vector quantization. The selection of LSVQ part of the LSGVQ is not very critical, similar values of SD being obtained for many different LSVQ stages.

6. REFERENCES

[1] K.K.Paliwal and B.S.Atal, "Efficient vector quantization of LPC parameters at 24 bits/frame," *IEEE Trans. Speech and Audio Processing*, vol.1, No.1, pp.3-14, Jan., 1993

[2] W.P.LeBlanc, B.Bhattacharya, S.A. Mahoud, V.Cuperman, "Efficient search and design procedures for robust multi-stage VQ of LPC parameters for 4 kb/s speech coding," *IEEE Trans. Speech and Audio Processing*, vol.1, No.4, pp.373-385, Oct., 1993

[3] J.Pan and T.R.Fischer, "Vector quantization of speech line spectrum pair parameters and reflection coefficients," *IEEE Trans. Speech and Audio Processing*, vol.6, No.2, pp.106-115, Mar.,1998

[4] A.Vasilache, M. Vasilache and I. Tabus, "Predictive multiple-scale lattice VQ for LSF quantization", In proc. In *ICASSP-99*, Vol.2, pp.657 –660, 1999

[5] M.Xie and J.-P.Adoul, "Fast and low-complexity LSF quantization using algebraic vector quantizer,", In *ICASSP-95*, vol.1, pp.716-719, 1995

[6] S.Ragot, J.-P.Adoul, R.Lefebvre, and R.Salami, "Low complexity LSF quantization for wideband speech coding," *IEEE Workshop on* Speech Coding Proceedings, pp. 22 –24, 1999

[7] A.Buzo, A.H.Gray, R.M.Gray and J.D.Markel, "Speech coding based upon vector quantization," *IEEE Trans. Acoust. Speech, and Signal Proc*ess., ASSP-28, pp.562-574, Oct. 1980shape gain VQ

[8] M.J.Sabin and R.M.Gray, "Product code vector quantizers for waveform and voice coding," *IEEE Trans. Acoust., Speech, Signal Processing*, vol.ASSP-32, no.3, pp. 474-488, June 1984

[9] A.Gersho and R.M.Gray, *Vector Quantization and Signal Compression*, Norwell, MA:Kluwer,1992

[10] J.H.Conway and N.J.A.Sloane, "Fast quantizing and decoding algorithms for lattice quantizers and codes," *IEEE Trans. Inform. Theory*, vol.IT-28, no.2, pp227-232, Mar. 1982

[11] R. Salami *et al*, "Design and description of CS-ACELP: a toll quality 8 kb/s speech coder," *IEEE Trans. Speech and Audio Processing*, vol.6, No.2, pp.116-130, Mar., 1998