SIMPLE FAST VECTOR QUANTIZATION OF THE LINE SPECTRAL FREQUENCIES

Jin Zhou, Yair Shoham
Bell Laboratories
Lucent Technologies
700 Mountain Ave, Murray Hill, NJ

Ali Akansu
Electrical Engineering Department
New Jersey Institute of Technology
Newark, NJ

ABSTRACT

In this paper we propose a simple fast-search VQ of the LSF's to be used on top of the split VQ, that is, in each of the sub-vector domains. The main trait of the proposed method is that no sub-optimal codebooks are used and there is no further reduction in performance. In each sub-vector domain, a full-size optimally trained codebook, typically of size 1024, is searched using a fast-search algorithm. The result of this search is identical to that of a full-search, yet, only about 25% of a full-search complexity is needed.

1. INTRODUCTION

Most of modern speech coding algorithms use the linear prediction (LP) parameters for describing and coding the speech spectral envelope [1]. The LP parameters are commonly transformed to line spectral frequencies (LSF) [2] for more efficient quantization and interpolation.

Various methods for coding and transmitting the LSF's have been studied. Earlier works were based on scalar quantization [2]. The demand for higher performance at lower bit rates has shifted the focus to the use of more powerful, yet more complex, vector quantization (VQ) techniques [3]. Typically, about 30 bits are assigned to coding 10-dimensional LSF vectors with a resulting spectral distortion of less than 1dB. However, 30-bit full-search VQ is totally impractical in terms of both computational complexity and memory space.

Various standard sub-optimal low-complexity VQ techniques have been used for coding the LSF's. The most commonly used method is the split VQ [3]. A 10-dimensional LSF vector is partitioned into typically three sub-vectors of size 3, 3 and 4. Each sub-vector is then independently coded. This method reduces the complexity and memory space significantly at the price of reducing the overall coding performance. However, unstructured full-search VQ of the LSF sub-vectors is still too complex for many applications.

The problem addressed in this paper is the reduction of the LSF split VQ complexity without a compromise in performance. The fast VQ methods proposed here are applied to each sub-vector independently and achieve the same level of performance at about 25% of the full-search split VQ complexity. The proposed method is based on classified VQ (CVQ) [4], originally used in image coding [5]. In classified VQ, shown in Figure 1, the input vector is first determined to belong to a certain class out of a pre-determined number of classes. Each class is represented by a small set of codevectors, which is a segment (C_i in Figure 1) in the optimal codebook. The union of all sets (classes) form the optimal codebook and the sets may overlap. Once the class is determined, a full-search is conducted over the set of that class and possibly over a few neighboring sets. The set to be searched is small, yet, the optimal codevector (the one selected by a full-search) is included (and selected) with extremely high probability. Note that CVQ requires about the same or even slightly more memory space since the fully optimal main codebook is used. The reduced-memory VQ problem is not addressed in this work.

The design of the classifier is the main issue in CVQ. Sophisticated classifiers partition the space very efficiently, resulting in small class sets. However, they require additional computations and memory. The expense of such classifiers may undo their benefit. In this paper we deal with 2 types of classifiers. The first is based on VQ. The second is based on simple feature extraction for which the added complexity is almost negligible.

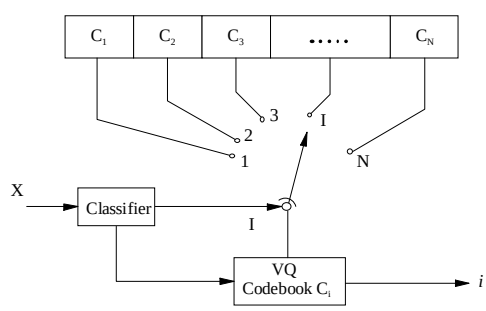

Figure 1: Classified VQ

Sections 2 to 4, describe the proposed classifiers and CVQ systems. The systems are characterized by complexity, memory and performance. The complexity unit is the effort required to compute a weighted distance per dimension. So, the complexity of a k-dimensional VQ of size N is $k*N$. The memory is given in terms of computer words. The performance is measured by average spectral distortion in dB [3]. Another performance figure is the percent number of miss-quantized vectors, namely, different from the ones obtained by full-search VQ. In this work, we assume a 3, 3, 4 split of the 10-dimensional LSF vector and a codebook of 1024 codevectors for each LSF sub-vector.

Extensions can obviously be made for different dimensions and sizes. Section 5 provides a comparative summary of simulation results for the proposed systems and concludes the paper.

2. CVQ WITH VQ-BASED CLASSIFIER

The classifier of this system is based on VQ. The optimal codebook is partitioned into *N* sets (classes) defined by *N* centroid vectors that are members of the optimal codebook. The codebook is rearranged such that a centroid and all its nearest neighbors occupy a contiguous segment of the codebook. An index table is used for pointing to centroids. The classifier uses this table to find *M* nearest centroid candidates (classes) to an input sub-vector. The corresponding sets are then searched for the final codevector. The complexity of this scheme is roughly $10*N+10*1024*M/N$. The added memory (index table) is $3*N$ words. Table 1 shows the performance trade-off for various *M*, *N* values.

3. LOW-COMPLEXITY CVQ WITH A MEAN-BASED CLASSIFIER

In this system, the classifier uses the mean of the sub-vector as a discriminating feature. The codebook is geometrically partitioned into classes by parallel hyperplanes as illustrated in Figure 2 for a two-dimensional case. To accomplish this, the codebook is rearranged in an order of increasing codevector means and then divided into *N* equal-size sets. *N* means of the 1st vector in each set are held in a class table. The classifier extracts the mean of the input vector and performs scalar quantization (SQ) using this table. The result of the SQ points to a certain set in the codebook. This set and $M-1$ of its closest neighboring sets (a total of *M* candidates) are searched for the final codevector.

The complexity of this scheme is $3*\log(N)+10*1024*M/N$. The 1st and 2nd terms are for the classifier and the quantizer, respectively. The added memory is $3*N$ words. The performance trade-off for various values of *M* and *N* is shown in Table 2.

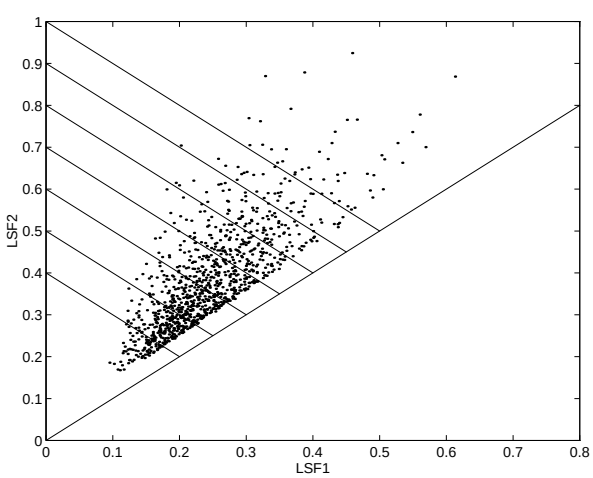

Figure 2: Mean-based Partitioning of the LSF Space

As an alternative way to dealing with a discrete number of disjoint classes, it is possible to search over a specified range around a location pointed to by the classifier. This is equivalent to having overlapping class sets. The need to maintain classes can be alleviated by putting all codevector means in the class table and performing SQ of size 1024. The classifier is now able to point to any codevector in the codebook. The codebook is searched within a windowed range of indices around the selected pointer. The added classifier complexity of $3*\log(1024)$ is still marginal.

The location of the window is determined by the profile of the ordered means. A typical curve is shown in Figure 3. To account for the curve nonlinearity, the window center can be dynamically shifted so that flatter regions (where the mean provides lesser discrimination) are better covered than the steeper ones. Let *I* be the pointed index and *W* the window width. The VQ search range is from $I-A(I)*W$ to $I+(1-A(I))*W$, $A(I)$ as a function of *I* should be found experimentally. Roughly, $A<0.5$, $A=0.5$ and $A>0.5$ for the regions marked by 1, 2, 3 in Figure 3, respectively.

The complexity of this scheme is $3*\log_2(1024)+10*W$ $=30+10*W$. The added memory is $3*1024$ words. The performance versus window size for this method is given in Table 3.

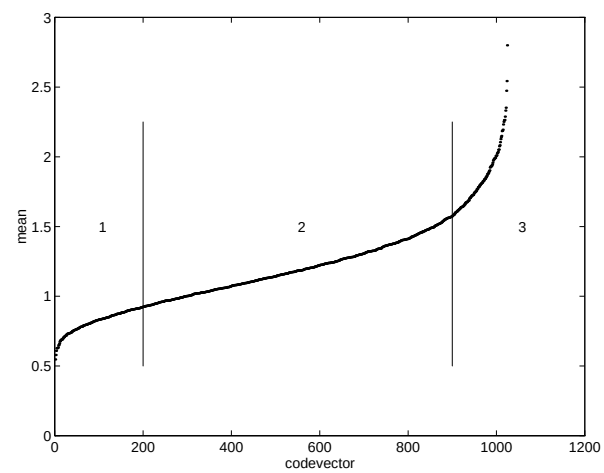

Figure 3: Ordered Means of LSF Codevectors

4. LOW-COMPLEXITY CVQ WITH A MIDDLE-COMPONENT CLASSIFIER

A trivial feature of a *k*-dimensional LSF sub-vector is the value of the middle component indexed by round (*k*/2). Since the LSF's are monotonically increasing, the middle component roughly follows the mean. The system is as in section 3 except that the middle component replaces the mean.

The geometrical interpretation of this method is the partitioning of the LSF space by hyperplanes that are all orthogonal to the

middle component axis, as shown in Figure 4 for the 2-dimensional case.

The advantage here is that no additional memory is needed since the middle components are in the codebook. It is therefore of interest to measure the discriminating power of this simple feature for classification purposes. The codebook is rearranged in an order of increasing middle components. As before, either the class table or the window methods can be used.

In the class table approach, N equally spaced codebook indices form an index set to a class table. Using this table, the middle component of the input vector is scalar quantized to select a segment (class) in the codebook. M neighboring classes are then searched for the final codevector. The complexity is $3*\log(N)+10*1024*M/N$. The added memory is zero. The performance trade-off for this scheme is given in Table 4.

In the window method, the SQ table includes 1024 middle components. The codebook search is done as described in section 3. The complexity for window width W is $3*\log(1024)+10*W$. The added memory is zero. The performance trade-off for this scheme is given in Table 5.

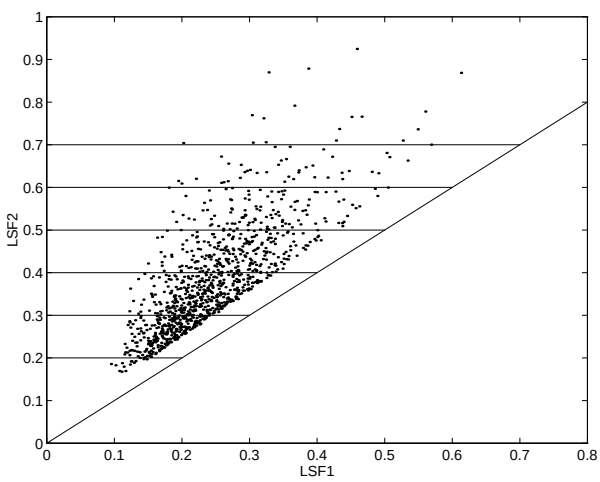

Figure 4: Second Component Classifier

5. PERFORMANCE OF THE PROPOSED SYSTEMS

Tables 1 to 5 show the performance trade-off between number of classes and number of candidates or window width for the systems described in section 4. For easy comparison, the complexity and added memory are expressed in percent of the full-search VQ respective values. The number of miss-coded vectors are given in percent of the test sequence size (20000 LSF vectors). A full-search 10-bit split VQ over the test sequence used in the experiment gives an average spectral distortion of 0.757 dB. This is the target performance of the proposed low-complexity systems.

Num. of Class	Performance	Number of Candidates				
		1	3	6	8	10
16	SD(dB)	0.889	0.764	0.757		
	Complex	10.1%	24.7%	44.9%		
	Missed	52.0%	4.38%	0.08%		
	Memory	0.47%	0.47%	0.47%		
32	SD(dB)	0.917	0.770	0.758	0.757	
	Complex	7.49%	15.4%	26.6%	33.7%	
	Missed	61.1%	9.55%	0.81%	0.17%	
	Memory	0.94%	0.94%	0.94%	0.94%	
64	SD(dB)	0.953	0.782	0.760	0.758	0.757
	Complex	8.5%	12.4%	18.1%	22.0%	25.5%
	Missed	71.2%	17.3%	1.71%	0.35%	0.1%
	Memory	1.88%	1.88%	1.88%	1.88%	1.88%

Table 1: CVQ with VQ Classifier

Num. of Class	Performance	Number of Candidates				
		1	3	5	9	17
16	SD(dB)	0.903	0.759	0.757		
	Complex	6.4%	18.8%	31.4%		
	Missed	62.8%	1.86%	0.11%		
	Memory	0.47%	0.47%	0.47%		
32	SD(dB)	1.113	0.793	0.761	0.757	
	Complex	3.27%	9.52%	15.8%	28.3%	
	Missed	86.5%	21.2%	3.31%	0.16%	
	Memory	0.94%	0.94%	0.94%	0.94%	
64	SD(dB)	1.505	0.939	0.816	0.763	0.757
	Complex	1.74%	4.86%	7.99%	14.2%	26.7%
	Missed	96.8%	64.3%	29.5%	4.9%	0.2%
	Memory	1.88%	1.88%	1.88%	1.88%	1.88%

Table 2: CVQ with Mean Classifier

Performance	Window Width				
	50	100	140	200	252
SD(dB)	0.918	0.785	0.764	0.758	0.757
Complex	5.18%	10.06%	13.97%	19.82%	24.90%
Missed	57.6%	16.35%	5.19%	0.78%	0.25%
Memory	30.0%	30.0%	30.0%	30.0%	30.0%

Table 3: CVQ with Mean Classifier - Window Method

Table 6 summarizes the performance of all the CVQ systems tested. It is shown that an average distortion of a full-search VQ can be achieved by low-complexity CVQ at only about 25% of the full-search complexity and with a very small miss-coding count. This is accomplished by the VQ and the mean-based classifier. The middle-component systems achieve the target performance with higher complexity but require no additional memory.

Num. of Class	Performance	Number of Candidates				
		1	4	7	13	26
16	SD(dB)	0.977	0.761	0.757		
	Complex	6.4%	25.2%	43. 8%		
	Missed	70.7%	2.04%	0.13%		
	Memory	0.47%	0.47%	0.47%		
32	SD(dB)	1.242	0.800	0.764	0.757	
	Complex	3.27%	13.6%	22.0%	40.8%	
	Missed	90.5%	19.0%	3.28%	0.17%	
	Memory	0.94%	0.94%	0.94%	0.94%	
64	SD(dB)	1.664	0.939	0.815	0.766	0.757
	Complex	1.74%	6.43%	11.1%	20.5%	40.8%
	Missed	97.9%	57.3%	24.6%	4.19%	0.15%
	Memory	1.88%	1.88%	1.88%	1.88%	1.88%

Table 4: CVQ with Mid-component Classifier

Performance	Window Width				
	60	100	200	310	382
SD(dB)	0.953	0.831	0.766	0.758	0.757
Complex	6.15%	10.1%	19.8%	30.6%	37.6%
Missed	60.02%	29.54%	4.39%	0.71%	0.20%
Memory	0	0	0	0	0

Table 5: CVQ with Mid-component Classifier - Window Method

	Complexity	SD (dB)	Missed Rate	Memory
VQ	25.4%	0.757	0.1%	1.875%
Mean Class	26.74%	0.757	0.195%	1.875%
Mean Window	24.90%	0.757	0.25%	30.0%
Middle Component Class	40.80%	0.757	0.145%	0.1875%
Middle Component Window	37.60%	0.757	0.195%	0

Table 6: Comparison of 5 methods

Figure 5 shows the distortion versus complexity curves for the tested systems. The use of these schemes for low-complexity speech coding depends on the best trade-off for the application in mind.

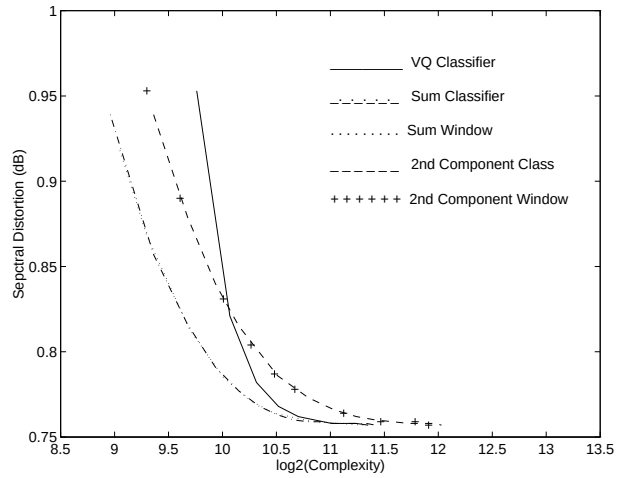

Figure 5: Spectral Distortion versus Complexity

REFERENCES

1. F. Itakura, "Linear spectrum representation of linear predictive coefficients of speech signals", *J. Acoust. Soc. Amer*., vol. 57, suppl. no. 1, pp. s35, 1975

2. F. K. Soong and B. H. Juang, "Line Spectrum Pair (LSP) and speech data compression", *ICASSP*, pp. 1.10.1, 1984.

3. K. K. Paliwal and B. S. Atal, "Efficient vector quantization of LPC parameters at 24 bits/frame", *IEEE Transactions on Speech and Audio Processing*, pp. 3-7, Jan., 1993.

4. A. Gersho and R. M. Gray, *Vector Quantization and Signal Compression*, Kluwer Academic Publishers, Norwell, MA, 1991.

5. B. Ramamurthi and A. Gersho, "Classified vector quantization of images", *IEEE Transaction on Communication*, pp.1105-1109, Nov., 1986.

6. F. K. Soong and B. H. Juang, "Optimal quantization of LSP parameters", *IEEE Transactions on Speech and Audio Processing*, pp. 15-19, Jan., 1993.

7. J. D. Markel and A. H. Gray, Jr., *Linear Prediction of Speech*, Springer-Verlag, Berlin, Heidelberg, New York, 1976.

8. L. R. Rabiner and R. W. Schafer, *Digital Processing of Speech Signals*, Prentice-Hall, Englewood Cliffs, NJ, 1978.

9. R. Hagen, *On Robust LPC-spectrum Coding and Vector Quantization*, Chalmers University of Technology, Goteborg, Sweden, 1995.