TRELLIS AND CODEBOOK SOURCE CODERS - A PERFORMANCE COMPARISON

M. Elbaz, S. Farkash & D. Malah

Department of electrical engineering
Technion - Israel Institute of Technology
Haifa 32000, Israel

Abstract

This paper presents a comparison between two vector source coders: codebook and trellis coders. The comparison, based on simulations, is referred to three aspects: performance, i.e., the SNR achieved by the coders, the amount of computations and the memory requirements. The results show that under the constraint of same amount of computations, and in the range the parameters were tested, there exists a trellis coder with better performance than the codebook coder. When the constraint is put on the available memory, then, beyond a critical memory size, a trellis coder can be constructed, which exceeds the performance of the codebook with a lower computational load. Guide-lines for setting the trellis parameters in order to design an efficient trellis coder are drawn.

I. Introduction

There are two basic approaches to implement source coding - scalar and vector coding [1,2]. A fundamental result of source coding theory states that better performance can always be achieved by vector coding rather then scalar. A bound on the achievable performance for any coder is given by the rate-distortion function R(D) representing the minimum achievable distortion when compressing data at any given rate for a given source [3].

A coder is said to be **optimal** if it's performance can reach the rate-distortion bound arbitrarily close. Three vector coders were proven to be optimal - the block coder (codebook), the tree coder and the trellis coder [1-3]. In this paper two of these coders, the codebook and the trellis coders are compared.

The most common vector coder is the codebook coder [1,2], which is constructed of an encoder and a decoder containing each a list of vectors referred to as a codebook. The main motivation in using the codebook coder hinges on its optimality, which is expressed by the fact that the codebook coder performance reaches the rate-distortion bound as the vector length goes to infinity. Two other important motivations in using the codebook coder are its conceptual simplicity, and the fact that an efficient (but not optimal) algorithm (LBG) for the construction of a codebook from a training sequence exists [4].

A major drawback of the codebook quantizer is its complexity. The number of code elements in the codebook increases exponentially with the vector length, implying exponential growth of the number of distortion measure computations and memory requirements. This fact prevents, in essence, the use of the codebook quantizer for long vectors.

The second vector coder dealt with in this paper is the trellis coder, which has recently attracted considerable interest [7-9]. The trellis coder, although not as conceptually simple as the codebook coder, was also proven to be optimal [3]. Furthermore, there is a linear dependence of the computational and storage requirements vs. the vector length, thus, implying lower complexity of the trellis coder. However, to the best of our knowledge, there is no thorough investigation of which of these two coders - codebook or trellis - is superior in performance under complexity (computation and memory) constraints.

In this paper we compare, on the basis of simulations, the codebook and trellis vector coders referring to three parameters: the performance, i.e., the distortion vs. rate; the amount of computations; and the memory requirements. The comparison is conducted for the rate of R = 1 bit per element. Similar qualitative results and conclusions are expected for other rates as well. It is shown that under the constraint of both coders having the same amount of computations, there exist a set of trellis parameters yielding a trellis coder with better performance than that achieved by the codebook coder. When the constraint is put on the available memory, then, beyond a critical memory size, a trellis coder can be constructed, which exceeds the performance of the codebook with a lower computational load.

Based on the comparison presented in the sequel, guide lines for setting the trellis parameters are established in order to achieve improved performance of the trellis coder. This design rules are important because of the many parameters and considerations involved in designing a trellis diagram.

The paper is organized as follows: Section II presents the codebook coder. Section III describes the trellis coder. In section IV expressions for the computation and memory requirements for the two coders are given. Section V summarizes the performance comparison. Guide lines for designing the trellis coder are drawn in chapter VI, and section VII concludes the paper.

II. The codebook coder

The codebook coder is a straight-forward solution to the vector coding problem [1-3]. In this coder both the encoder and decoder contain an identical list of M vectors of size N referred to as the codebook. The encoder assigns to each input vector of length N an index which corresponds to the nearest code vector in the codebook according to a prescribed distortion measure. This index constitutes the compressed data. The decoder uses the same index to fetch a code vector from its codebook. This code vector constitutes the reconstructed output vector.

When coding vectors of length N, using R bits per element, with a codebook containing M vectors, the following relations hold:

$$M = 2^{NR} \quad \rightarrow \quad R = \frac{1}{N}\log_2 M \qquad (1)$$

The codebook coder is proven to be optimal [3], since as

N goes to infinity the coder performance approaches the rate distortion bound.

The main problem concerning codebook coders is the determination of the codebook vectors. Motivated by the proof of the vector source coding theorem, the codebook can be randomly populated, using the optimal-coder output distribution law [3]. Another way is to use a training sequence from the source data and an iterative efficient algorithm such as the LBG [4].

III. The trellis coder

Both the encoder and decoder of the trellis coder are composed of a trellis diagram which plays the same role played by the codebook in the codebook coder. The trellis diagram (e.g., see Fig. 1) is a directed graph composed of nodes and branches. Each branch in the diagram is populated by n elements, and a set of N = nL elements along a path through the diagram constitutes a code vector, where L is the number of stages in the trellis (L=5 in Fig. 1).

The coding process is performed as follows:

For each input source vector the encoder searches the trellis for the optimal path, i.e., the path along the trellis which minimizes the distortion between the input source vector and the code vectors residing on paths through the trellis. The path-map corresponding to the chosen path represents the compressed data. The decoder uses the same path map in an identical trellis to uniquely determine the reconstructed output vector.

The structure of the trellis diagram is determined by a finite state-machine realized by a shift-register. For a shift register of size K with a radix-q alphabet, the number of nodes(states) in each stage of the diagram is q^{K-1}, and the number of branches emanating from each node (branching factor) is q. the connections between nodes at successive stages in the diagram correspond to the transitions of states in the shift register. The trellis diagram in Fig. 1 corresponds to q = 2 , K = 3 , and has L = 5 where L. The initial state of the shift register can be an arbitrary one, yielding a *full trellis*. Throughout this paper the initial state is chosen to be zero, resulting in a *simple trellis* such as the one depicted in Fig. 1.

When coding vectors of length N using a simple trellis coder, the rate R in bits/element is given by :

$$R = \frac{L}{N}\log_2 q = \frac{1}{n}\log_2 q \qquad (2)$$

An exhaustive search for the optimal path in the trellis requires a large amount of computations. However, use of the Viterbi algorithm [5] enables an efficient search for the optimal path.

The trellis coder is an optimal coder (in the sense discussed in section I) since as q or K go to infinity the coder performance approaches the rate distortion bound [3].

Similar to the codebook coder, a major issue is the population of the trellis diagram branches. As with the codebook coder, one can populate the diagram randomly - based on the proof of the trellis source-coding theorem [3]. In that case, the trellis is populated using the optimal-coder output distribution law. Applying an iterative training method such as the LBG algorithm is quite difficult here because of the trellis structure constraints. Several algorithms to populate the trellis diagram using a training sequence from the source data are found in the literature [6]. Since these algorithms as well as the LBG algorithm are not optimal, we have chosen to use random population for both the trellis and the codebook coders.

IV. Computation and memory requirements

Denoting the amount of computations per element as C(cb) - for the codebook, and C(tr) - for the trellis; and the memory size required as M(cb) and M(tr), respectively, the following expressions are obtained (for the simple trellis):

$$C(cb) = 2^{NR} \qquad (3)$$

$$M(cb) = N\cdot 2^{NR} \qquad (4)$$

$$C(tr) = \frac{(L+1-K)}{L}q^K + \frac{1}{L}\sum_{i=1}^{K-1} q^i \underset{L>>K}{\rightarrow} q^K \qquad (5)$$

$$M(tr) = n(L+1-K)q^K + n\sum_{i=1}^{K-1} q^i \underset{L>>K}{\rightarrow} Nq^K \qquad (6)$$

Note that using (2) we also have $q^K = 2^{nKR}$. The computation expressions above concern only distortion computations. The contribution of the comparison operations is not taken here into account.

First, we refer to the effect of the vector length on the memory and computation requirements. As can be seen from eqns. (3) and (4) both the computation and the memory requirements for the codebook coder are exponentially growing with the vector length N. The computation requirements for the trellis coder (eqn. 5) is independent (for sufficiently large L) on the vector length, whereas the memory required for the trellis coder (eqn. 6) is linearly growing with the vector length. This behavior may indicate possible trellis coder superiority regarding memory and computation requirements vs. vector length applied.

Next we examine the computation and memory requirements vs. rate. It can be seen from the above expressions that both are growing exponentially with increasing rate, but the exponent coefficient in the trellis expression is nK while for the codebook coder the coefficient is N. Since usually L>>K we have N = nL>>nK which means that the trellis coder can work with much longer vectors for the same rate, and same computations or memory requirements - again indicating possible trellis superiority.

V. Performance comparison

Since except for the rate-distortion bounds there are no analytical derivations for the performance of the codebook and the trellis coders, their performance comparison has been made on a basis of simulations. The simulations were done for the most common source - the white Gaussian source $\mathcal{N}(m,\sigma^2)$ with zero mean (m=0) and unity variance ($\sigma^2 = 1$).

For this source, the optimal coder output distribution low is $P(y) = \mathcal{N}(0,\sigma^2-D)$ (where D is the distortion determined form the rate distortion function R(D) for the given

rate), and this distribution law was used to randomly populate both the codebook and the trellis diagram. All the simulations were done for the rate of R = 1 bits/element. For this rate the rate-distortion bound on the achievable performance for the above source is 6.02 dB. In addition, the trellis length L was chosen to satisfy L>>K , as is commonly used.

Fig. 2 illustrates the performance vs. computations for both the codebook and trellis coders. The solid curves in the graphs correspond to the codebook coder, for which the amount of computations is varied by varying the vector length N according to (3). For the trellis coder the amount of computations is controlled by varying q (in Fig. 2a) or K (in Fig. 2b) according to eqn (5). The "+" signs correspond to vector length values of N = 10nK (i.e., L = 10K) whereas the "x" signs corresponds to a vector length value of N = 256 (maintaining L ≥ 10K).

The following conclusions can be drawn from these two graphs:

(i) There exists a set of trellis coder parameters yielding better performance then that obtained by the codebook coder for any given constraint on the computational load in the range of examined values.

(ii) The performance of the trellis coder is almost independent of the vector length for a given computational constraint if L ≥ 10K .

(iii) For a fixed computational load q^K = const, the performance of the trellis coder is slightly improved as K increases.

The improved performance of the trellis coder over the codebook coder for a given constraint on the computational load is achieved at the cost of increased memory needed for the trellis coder. This is given from the result that for the same computational load and for L>>K

$$M(tr) = M(cb)\frac{L}{K} \qquad (7)$$

Fig. 3 illustrates the performance versus memory size for both the codebook and trellis coders. Again, the solid curves in the graphs correspond to the codebook coder, for which the memory size is varied by varying the vector length N - according to (4). For the trellis coder the memory size is controlled by varying q (in Fig. 3a) or K (in Fig. 3b) according to eqn (6). As in Fig. 2, the "+" signs correspond to vector length values of N = 10nK (i.e., L = 10K), whereas the "x" signs corresponds to a vector length value of N = 256 (maintaining L ≥ 10K).

The following conclusions can be drawn from Fig. 3:

(i) If memory size is increased while keeping q^K fixed (i.e., increasing N), the performance of the trellis coder is only slightly improved, as indicated by the dashed lines on the graphs.

(ii) The trellis coder performance is improved as q^K increases (but maintaining L ≥ 10K. Furthermore, for the range of vector length values examined, there exists a *threshold* value for q^K (128 in Fig. 3) above which the trellis coder performance exceeds the performance of the codebook coder. This threshold value for q^K implies a *critical memory size* above which there exist q and K values yielding a trellis coder having a better performance than the codebook coder. Note however, that further increasing the memory size while maintaining q^K fixed (i.e., increasing the vector length N) will lead eventually to superiority of the codebook performance (at the price of exponentially increasing computations), as it asymptotically reaches the rate distortion bound with increasing N, while the trellis coder performance can not reach this bound with finite values of q and K , even as N goes to infinity.

From eqns. (4) and (6), if both coders have the same memory size then

$$\frac{C(tr)}{C(cb)} = \frac{N(cb)}{N(tr)} \qquad (9)$$

where, N(tr) and N(cb) are the vector length of the trellis and the codebook coders, respectively. Hence if the memory size exceeds the critical value discussed above, there exist a trellis coder with equal or better performance and a lower computational load as compared to the codebook coder. In the case shown in Fig. 3, at the point where both coders have the same performance: C(tr) = C(cb)/7 .

VI. Guide lines for designing trellis coder

In this section we draw guide-lines for designing the trellis parameters, using Figs. 2 and 3 and the conclusions above. Recall that the simulations and the analysis were done only for the rate R = 1 bits/element. However, we conjecture that similar qualitative results will be obtained also at other rates. Two possible approaches to the problem of designing a source coder are considered, depending on the constraint put on the coder are considered here:

(A) Constraint on the amount of computations - C_c

1. For a given rate R, the integer parameters q and n should be chosen such that $q = 2^{nR}$ has the lowest possible value.
2. Given the constrained amount of computations C_c, choose the largest possible value of K satisfying $q^K \le C_c$.
3. The number of stages L in the trellis should be chosen such that L>>K, yielding vector length N = nL.

(B) Constraint on the available memory - M_c

Assuming that the critical memory size for a given rate is known, and the memory constraint, M_c , exceeds this critical value, the following design rules are suggested.

1. For a given rate R, the integer parameters q and n should be chosen such that $q = 2^{nR}$ has the lowest possible value.
2. Given the constraint on the memory size M_c, choose the largest possible value of K satisfying $Kq^K << M_c/n$ (to satisfy a requirement of the form a<<b we recommend here a ratio b/a ≥ 10).
3. The number of stages L in the trellis should be chosen the largest possible value satisfying $L \le M_c/(nq^K)$.

VII. Summary and Conclusions

A comparison between two optimal vector source coders - the codebook coder and the trellis coder - has been presented. The comparison between the trellis and the codebook was performed on a basis of simulations using a Gaussian source and the squared-error distortion

measure. The comparison was performed for rate R = 1 bit per element, Similar qualitative results and conclusions are expected for other rates as well, using the same method.

It was found that for a given constraint on the computations there exists a trellis coder which yields a better performance than the codebook coder. When the constraint was put on the memory size, it is found that there is a critical memory size above which a trellis coder can be constructed having the same or better performance then the codebook coder, yet with a lower computational load. Similar qualitative results were obtained for Laplace distribution source using the absolute distortion measure, but are not presented here because of lack of space.

A set of design rules is given to guide the design of an efficient trellis coder. These guide lines are given for two basic constraints: memory size and computations.

REFERENCES

[1] R. Gray, "Vector Quantization", IEEE ASSP Magazine, Vol. 1, April 1984, pp. 4-29.

[2] N.S. Jayant and P. Noll, "DIGITAL CODING OF WAVEFORMS", Englewood Cliffs, NJ: Prentice-Hall, 1984.

[3] A.J. Viterbi and J.K. Omura, "PRINCIPLES OF DIGITAL COMMUNICATION AND CODING", McGraw-Hill, 1979.

[4] Y. Linde, A. Buzo and R.M. Gray, "An Algorithm for Vector Quantizer Design", IEEE Trans. Commun. Vol. COM-28, January 1980, pp. 84-95.

[5] G.D. Forney Jr., "The Viterbi Algorithm", Proc. IEEE, Vol. 61, March 1973, pp. 268-278.

[6] M.W. Marcellin and T.R. Fischer, "Trellis Coded Quantization of Memoryless and Gauss-Markov Sources", IEEE Trans. Commun., Vol. 38, No. 1, January 1990, pp. 82-93.

[7] B. Mazor and W.A. Pearlman, "A Trellis Code Construction and coding Theorem for Stationary Gaussian Sources", IEEE Trans. Inform. Theory Vol. IT-20, No. 3, May 1983, pp. 924-930.

[8] B. Mazor and W.A. Pearlman, "An Optimal Transform Trellis Code With Application to Speech", IEEE Trans. Commun., Vol. COM-31, No. 6, June 1983, pp. 835-839.

[9] S. Farkash, D. Malah and W.A. Pearlman, "Transform Trellis Coding of Images at Low Bit Rate", IEEE Trans. Commun., Vol. COM-38, No. 10, Oct 1991, pp. 1871-1878.

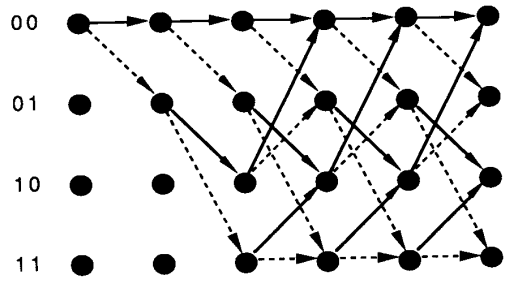

Fig. 1 - The trellis diagram for Q = 2 , K = 3 and L = 5
(pass bit-map: solid - '0'; dashed - '1')

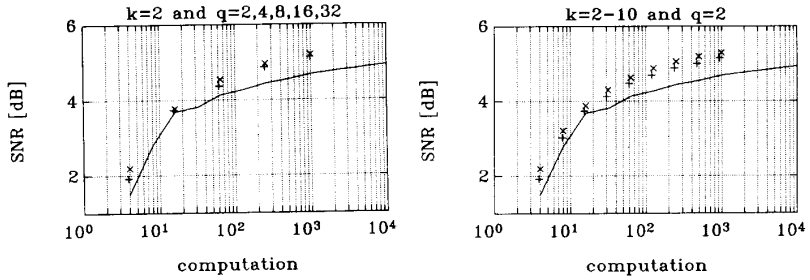

Fig. 2 - Coders performance (SNR) vs. computations (Gaussian source)
solid line - codebook coder, +,x - trellis coder
(a) Fixed K, varying q. (b) Fixed q, varying K

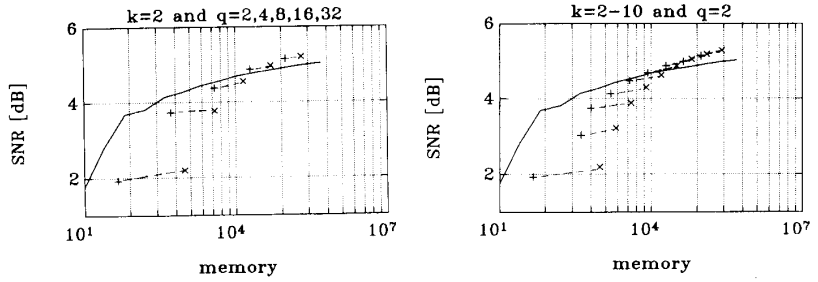

Fig. 3 - Coders performance (SNR) vs. memory (Gaussian source)
solid line - codebook coder, +,x - trellis coder
(a) Fixed K, varying q. (b) Fixed q, varying K