Soft-Decision Decoding of Fixed-Rate Entropy-Coded Trellis-Coded Quantizer Over a Noisy Channel

Sasan Nikneshan, *Member, IEEE*, Amir K. Khandani, *Member, IEEE*, and Peter Kabal, *Member, IEEE*

***Abstract*—This paper presents new techniques to improve the performance of a fixed-rate entropy-coded trellis-coded quantizer (FE-TCQ) in transmission over a noisy channel. In this respect, we first present the optimal decoder for a fixed-rate entropy-coded vector quantizer (FEVQ). We show that the optimal decoder for the FEVQ can be a maximum likelihood decoder where a trellis structure is used to model the set of possible code words and the Viterbi algorithm is subsequently applied to select the most likely path through this trellis. In order to add quantization packing gain to the FEVQ, we take advantage of a trellis-coded quantization (TCQ) scheme. To prevent error propagation, it is necessary to use a block structure obtained through a truncation of the corresponding trellis. To perform this task in an efficient manner, we apply the idea of tail biting to the trellis structure of the underlying TCQ. It is shown that the use of a tail-biting trellis significantly reduces the required block length with respect to some other possible alternatives known for trellis truncation. This results in a smaller delay and also mitigates the effect of error propagation in signaling over a noisy channel. Finally, we present methods and numerical results for the combination of the proposed FEVQ soft decoder and a tail-biting TCQ. These results show that, by an appropriate design of the underlying components, one can obtain a substantial improvement in the overall performance of such a fixed-rate entropy-coded scheme.**

***Index Terms*—Combined source and channel coding, error propagation, fixed-rate entropy-coded vector quantization (FEVQ), trellis-coded quantization (TCQ), Viterbi algorithm.**

I. Introduction

WIRELESS communication, often characterized by narrowband and noisy channels, is getting a good deal of attention in multimedia application. Design principles stemming from Shannon source and channel-separation theorem are being reconsidered and attention is moving toward the joint source-channel coding and decoding as viable alternatives for reliable communications across noisy channel. First attempts at joint source-channel decoding considered the fixed-rate encoders. However, the wide use of variable-length codes in data compression has motivated recent consideration of their joint source-channel coding. Variable-length codes have an inherent problem in the presence of channel error. Due to the sequential decoding and variable-length nature of such codes, channel errors can lead to a loss of synchronization resulting in error propagation. In practice, errors may propagate for a considerable period of time before synchronization is reestablished. Variable-length codes (e.g., Huffman codes) might be improved in respect of synchronization as explored in [2] and [3]. The mechanism relies on the existence of universal synchronizing code words that will obviously result in some overhead in terms of the required bit rate. Conventionally, variable-length bit streams are made channel robust through packetization and forward error correction (FEC), which may result in an undesirable overhead in terms of bandwidth efficiency. Another problem with variable-length codes is that they require buffering for transmission over a channel. In practice, such a buffer has a finite size; hence, undesirable buffer overflow/underflows may occur. Recently, several joint source/channel coding approaches have been introduced to improve the decoding of variable-length codes.

One class of source-channel coders use some knowledge of the source or source coder properties to detect channel errors and/or compensate for the effect of such errors. In this case, no redundancy is added to the output of the source encoder. Instead, the characteristics of the source or the source code are used to provide protection against possible channel errors. For example, if there is some redundancy remaining at the output of the source encoder, then this residual redundancy can be used to combat the channel noise [10]. The application of such techniques for the case of variable-length encoded data are proposed by [4] and [6]–[8]. This technique combined with other error-correcting schemes, such as turbo codes, are also presented in [11]–[13].

In general, the nature of variable-rate systems greatly complicates the estimation problem at the decoder side and there have been different attempts to reduce the receiver complexity at the cost of being suboptimum. In [4], assuming the source to be first order Markovian, a maximum *a posteriori* (MAP) decoding method is presented, using an approximate method in which the receiver operation is independent of the probability of the received code word. In [5] and [6], a computationally complex exact MAP decoding method and an efficient approximation for it are studied. In addition to sequence-based MAP decoding, the symbol-by-symbol MAP decoding according to the BCJR algorithm [9] is also described. In [7], another method is proposed,

Manuscript received August 26, 2001; revised November 26, 2002 and June 25, 2003. This work was supported by the Communications and Information Technology Ontario (CITO) and the Natural Sciences and Engineering Research Council of Canada (NSERC). A preliminary version of this work is presented in [1].

S. Nikneshan is with RFTune, Inc., Waterloo, ON N2K3T6, Canada.

A. K. Khandani is with the Electrical and Computer Engineering Department, University of Waterloo, Waterloo, ON N2L 3G1, Canada.

P. Kabal is with the Electrical and Computer Engineering Department, McGill University, Montreal, QC H3A 2A7, Canada.

Digital Object Identifier 10.1109/TVT.2004.823482

specifically for memoryless sources. In [8], a MAP-decoding method that does not include a constraint on the length of the decoded symbol sequence is proposed.

To take advantage of the potential gain due to entropy coding while avoiding the disadvantages associated with conventional methods based on using variable-rate codes (including error propagation and buffering problems), one can use a fixed-rate entropy-coded vector quantizer (FEVQ), which is derived from a set of variable-length scalar quantizers. The use of FEVQ confines the error propagation within a block, resulting in better performance over noisy channels. As we will show in this article, by using residual redundancy in FEVQ, one can further improve the overall performance over a noisy channel. The methods presented in this paper exploit the known characteristics of the output symbols of an FEVQ to provide higher protection against the channel noise, resulting in reduction of the error propagation between the symbols within an FEVQ code vector.

In an FEVQ, every code vector consists of a fixed number of bits representing N source symbols [14]–[16]. Although an FEVQ attempts to remove the redundancy of the source in the first place, due to the imposed constraint on having a fixed number of bits per block of N source symbols, this objective of redundancy removal cannot be achieved completely. In this article, the receiver is designed to take advantage of such a leftover redundancy in the encoded source output to improve the overall performance of the decoder in handling possible channel errors. We show that the optimum receiver can be a MAP or maximum likelihood (ML) receiver. In this case, the decoder estimates the transmitted sequence by applying the Viterbi algorithm through a trellis structure (finding the most probable sequence) where the trellis is a model for representing the entire encoder code book.

It is also well known that the choice of mapping between the quantizer code words and channel input symbols may lead to a reduction in the distortion due to channel noise [17], [18]. This is known as the index-assignment problem. The idea here is to come up with a distance preserving mapping from the source space to the channel space such that a "small" channel noise results in a "small" source distortion. In this regard, we propose a procedure to label the prefix code tree to improve the overall performance.

In order to achieve some quantization packing gain, we make use of a trellis-coded quantizer (TCQ) in conjunction with FEVQ. To prevent the effect of error propagation, it is necessary to use a block structure obtained by a truncation of the trellis. In TCQ, there are two common methods to implement such a block structure, either by transmitting an additional number of bits to specify the starting state for each block or by starting the TCQ operation in a fixed starting state for each block. Both methods suffer from an overhead in terms of either the required bit rate or the achievable quantization gain. The disappointing fact is that, as the number of states increases, the corresponding loss substantially increases in both methods. To reduce the effect of such a loss, we propose the use of a new form of block-structure TCQ by the application of a tail-biting trellis. The use of tail-biting trellis significantly reduces the required block length with respect to conventional methods, which results in a smaller delay and also mitigates the effect of the error propagation in signaling over a noisy channel. We will also study the combination of the tail-biting TCQ with the proposed ML decoder of the FEVQ.

The rest of the paper is organized as follows. Section II discusses the soft decoding of the FEVQ over a noisy channel. In this regard, we have a brief review of the FEVQ formulation where the FEVQ is based on a variable-length scalar quantizer labeled with a binary prefix code. We show that the optimum receiver of the FEVQ can be an MAP or an ML decoder. A trellis representation of the FEVQ decoder is then described. We further show how the idea could be generalized for the case that FEVQ is combined with TCQ. We conclude in Section II by presenting a procedure to label a prefix code tree. Section III introduces the tail-biting trellis structure for the quantization and a suboptimum algorithm is also presented for the decoding of the proposed tail-biting TCQ. Finally, in Section IV, we conclude the paper with some numerical results regarding the soft decoding of FEVQ, tail-biting TCQ, and the combination of the two.

II. Soft Decoding of FEVQ Over a Noisy Channel

A. FEVQ Structure

Consider an N–dimensional vector quantizer derived from N variable-length scalar quantizers. The ith quantizer consists of M_i partitions with reconstruction levels $\{r_i(1), r_i(2), \ldots, r_i(M_i)\}$, where $r_i(1) < r_i(2) < \cdots < r_i(M_i)$. There is a variable-length binary prefix code $C_i = \{c_i(1), c_i(2), \ldots, c_i(M_i)\}$ associated with each quantizer, where the code word corresponding to $c_i(j)$ has a length of $\ell_i(j)$. Using the above notations, the FEVQ operation can be formulated as

$$\text{minimize} \quad \sum_{i=1}^{N} (x_i - r_i(j))^2$$
$$\text{subject to} \left\{ \sum_{i=1}^{N} \ell_i(j) \le L_{\max} \right. \tag{1}$$

where $L_{\max}$ is the maximum binary length to represent an N-dimensional (N-D) code word. The output of the FEVQ will be a bit sequence with binary length of $L_{\max}$. The formulation differs from [15], where the self-information of the reconstruction levels is used instead of their binary code lengths. In case the total binary length to represent the N quantized symbols is less than $L_{\max}$, the rest of the block is filled by zeros. There are a few methods known to solve (1), providing a range of tradeoffs between performance and complexity for different classes of sources [14]–[16], [19].

B. ML Decoding of FEVQ

Assume that the FEVQ operates at a rate of R bits per source dimension, i.e., $L_{\max} = RN$. For each input, the encoder produces a binary vector $\mathbf{X} \in \{0,1\}^{RN}$ for transmission. Each of the RN bits of $\mathbf{X}$ is binary phase-shift keying (BPSK)

modulated and the output $\mathbf{Y} \in \{-1, 1\}^{RN}$ is transmitted over an additive white Gaussian noise (AWGN) channel receiving an N-D real vector Z.

To recover the transmitted code-word sequences from the received data Z, the straightforward approach is to decode in a sequential manner (symbol by symbol). This is achieved by hard-decision decoding of bits and then reverse substitution after parsing the decoded string into code words. In the case that the received bits are not enough to specify all the N symbols (error has occurred), the decoder maps the rest of the symbols to a fixed value equal to the statistical average of the reconstruction levels. In this method, the receiver uses neither the fact that each block contains N source symbols nor the information that is embedded in the soft data.

Using a received vector Z, the optimum receiver [20] chooses the most probable transmitted sequences

$$P(X|Z) = \frac{P(Z|X)P(X)}{P(Z)}.$$

For fixed-length codes, $P(Z)$ is irrelevant to the receiver operation and the optimal receiver maximizes $P(Z/X)P(X)$. This assumption is not valid for the variable-length codes (the received vectors Z have different probabilities) and, in related research works, in order to simplify the decoder operation, the effect of $P(Z)$ is ignored [4], [6], [8]. For an FEVQ encoder, all the transmitted sequences (N-tuples) have the same binary length and almost the same probability (based on the asymptotic equipartition property (AEP) [21]). Therefore, the assumption that $P(Z)$ is approximately the same for all the transmitted code vectors is valid and the receiver can be simplified to a MAP decoder that maximizes

$$P\left(\frac{Z}{X}\right) P(X).$$

Once again, considering the fact that all encoder code vectors are approximately equiprobable (based on AEP), the receiver can be further simplified to an *ML decoder* that maximizes $P(Z|X)$. Thus, the decoder can be a well-known Viterbi decoder that uses $P(Z|X)$ as the path metric in a trellis representation of the FEVQ where the corresponding branch metrics depend strictly on the channel output. In the following, we further discuss the trellis structure used in the soft-decision decoding of the FEVQ.

C. Trellis Representation

Using the fact that each block of RN bits represents N source symbols, the decoder compares the received block with all the possible choices of N symbols that has a total binary length of $L_{\max}$ or less. One structured approach is to represent all of the combinations of N symbols by a trellis diagram, with the states corresponding to the accumulative length of the code words. In this trellis diagram, each transition corresponds to a set of source symbols, which results in a trellis composed of $N + 1$ stages, where the transition(s) from state s_j of stage $k - 1$ to state $s_{j+\ell}$ of stage k correspond to the kth symbol(s) of length ℓ (refer to Fig. 1). The states in the kth stage $k = 0, \ldots, N + 1$ represent the accumulative code-word length over the set of the first k dimensions. In this case, since the prefix code may consist of code words of equal length, there may exist parallel branches between some of the states. As we will see later, this is an important point to be noted in selecting the index assignment for increasing robustness in signaling over a noisy channel.

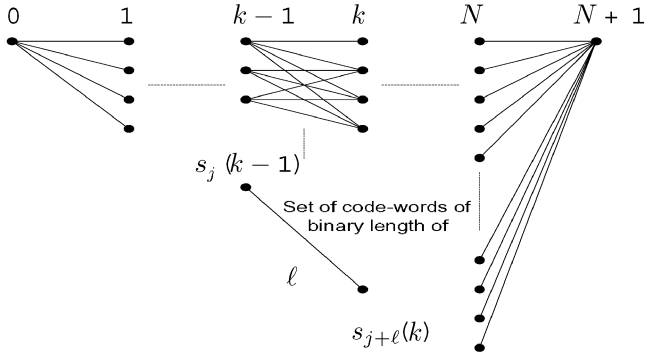

Fig. 1. Trellis structure for decoding of prefix codes.

In the final portion of the trellis (from stage N to $N+1$), there is a transition corresponding to the sequence of terminating zero bits, ending to a common final node for all the trellis paths. This ensures that the total length of all the paths is equal to $L_{\max}$. The final step is to use the Viterbi algorithm to find the most likely path through this trellis.

D. Soft Decoding of FE-TCQ

Consider an FE-TCQ specified by $\nu = 2^{\mu}$ states, an alphabet (set of quantization levels) $Q = \{q_1, q_2, \ldots, q_{2M}\}$, which is partitioned into four subsets S_0, S_1, S_2, and S_3 using an Ungerboeck partition chain, a set of binary code-word lengths $L = \{\ell_1, \ell_2, \ldots, \ell_{2M}\}$, where ℓ_i is the binary length corresponding to q_i, a threshold for total binary length of $L_{\max}$, and an Ungerboeck-type trellis structure $T(Q)$ whose branches are labeled with the subsets $S_i, i \in \{0, 1, 2, 3\}$. The code-book set is a collection of all the possible sequences from the trellis $T(Q)$, with an additional constraint that the total binary length is no greater than the threshold $L_{\max}$.

The ML decoder explained in the previous section can be applied for soft-decision decoding of the suggested FE-TCQ, with some differences caused by the method used to design the corresponding prefix code. We consider two possible cases to design such a prefix code.

I) A variable-length binary prefix code is assigned to the points of each of the Ungerboeck-type subsets S_0, S_1, S_2, S_3 [22]. In this case, every bit sequence corresponding to a threshold point consists of two parts: one bit that represents the subset[1] (TCQ path indicator) and the rest of the bits that represent the prefix code word. Since these two parts are separable in this method of labeling, the decoder could consist of an ML decoder (with the same trellis structure explained in the previous section) for the fixed-rate entropy-coded part plus a standard Viterbi decoder

[1]Note that following an Ungerboeck-type trellis structure for TCQ, at each node of the trellis, two of the possible four subsets are allowed.

for the TCQ part (the issue of TCQ decoding will be explained later).

II) A variable-length binary prefix code is assigned to $S_0 \cup S_2$ and $S_1 \cup S_3$ [23]. In this case, the path-indicator bits of TCQ and the prefix code-word bits are not separable. Therefore, the ML decoder for the fixed-rate entropy-coded part and the TCQ decoder are combined. In other words, the two trellis structures are merged and, as a result, the decoder complexity increases substantially.

E. Index Assignment

As discussed earlier, the FEVQ under consideration relies on a variable-length scalar quantizer with prefix codes. As prefix codes can be represented by a tree structure, the number of possible choices for different index assignments are limited. This makes the task of optimizing the index assignment much simpler in comparison with that of other vector quantizers. To obtain the best performance in assigning the binary code words to the quantization levels, we follow a rule that the quantizer points that have the *same binary length* and are far from each other should differ in as many bit positions as possible and the ones that are close together should differ in as few bit positions as possible. In other words, among the code words with equal binary lengths, the largest Hamming distance between the code words (indexes) should correspond to the largest Euclidean distance between the corresponding threshold levels and vice versa. This will reduce the chance of error between parallel branches in the trellis.

In a given prefix code (Huffman code), the tree can be labeled in different ways, resulting in different binary prefix codes, all with the same set of code-word lengths. Satisfying the above-mentioned rule is not possible for some choices of labeling. Our goal is to devise a method to label a binary tree such that it allows us to map indexes to reconstruction levels in accordance with their Euclidean distances from each other. The labeling of the tree is done such that the leaves of the tree from the left to the right represent the reconstruction levels of the scalar quantizer from the left to the right, respectively.

Assume there is a prefix code assigned to a set of reconstruction levels and the lengths of the code words and the binary tree representing the code words are known. For most class of sources e.g., Gaussian, Laplacian, etc., the probability density function (pdf) is a symmetric function with respect to the y axis $(x = 0)$. For these types of sources, the pdf is also usually monotonically decreasing in each side of the y axis. We assume that the source under consideration has such a pdf. To reflect the source symmetry in the prefix code tree, we choose the labels of one side of the tree to be a complement of the other side. This guarantees that code words of equal length get the maximum Hamming distance when representing two symmetrical points at the two sides of the y axis. Code words of equal length may also happen at one side of the y axis. Noting the assumption that the source pdf is monotonically decreasing, we conclude that such code words of equal length will correspond to neighboring quantizer partitions. In this case, to satisfy being a distance-preserving mapping, we choose the labels of one side of the tree

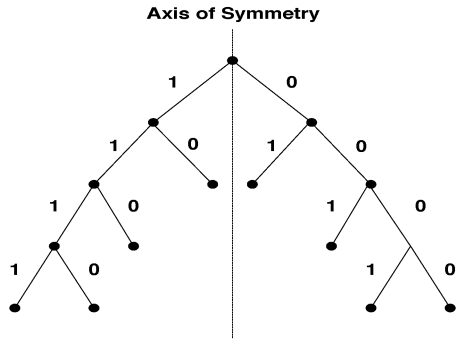

Fig. 2. Example of index assignment for a symmetric tree [Code(I) from Table I].

TABLE I
DIFFERENT INDEX ASSIGNMENT FOR PREFIX CODE STRUCTURES

Quantizer level	Code(I)	Code(II)	Code(III)
–2.17	0000	0100	0111
–1.36	0001	0101	0110
–0.77	001	011	010
–0.25	01	00	00
+0.25	10	11	10
+0.77	110	100	110
+1.36	1110	1010	1110
+2.17	1111	1011	1111

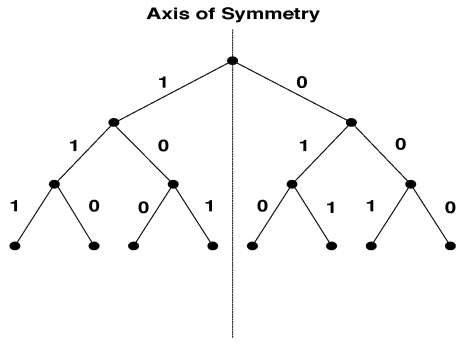

Fig. 3. Example of index assignment for a symmetric tree.

to form a gray code at any given depth, which guarantees that the neighboring labels differ in only one bit position. In order to have a gray code at any depth, we alternately map the $\{0, 1\}$ and $\{1, 0\}$ to subsequent pairs of leaves. In other words, if we have mapped the $\{0, 1\}$ to the first pair of leaves, the next pair will have the label of $\{1, 0\}$ and vice versa. (Obviously, this method does not result in a unique way of labeling.)

For example, Fig. 2 shows a symmetric binary tree representing Code(I) from Table I. The labeling in half of the tree (the right side) is done such that the code words form a gray code at any depth and the labels of one half of the tree are the complements of the other half. In Fig. 3 at depth 3, code words 000, 001, 011, and 010 form a gray code. In Fig. 3, in the right side of the tree at the depth of 3, there are two pairs of leaves that have the label of 0,1,1,0 from the left to the right. The following procedure describes how to label a given tree to satisfy the index-assignment conditions.

1) Index the right (left) side of the symmetric tree. The indexing is such that, at any depth, the code words form a gray code.

TABLE II
DIFFERENT INDEX ASSIGNMENT OF PREFIX CODE FOR FE-TCQ

Quantizer level	Code(I)	Code(II) (a)	(b)	(c)
–5.12	0 111	0111	0000	111111
–3.09	0 111	0111	0000	111110
–2.14	1 111	1001	0001	11110
–1.51	1 111	1001	0001	1110
–1.06	0 10	010	001	110
–0.69	0 10	010	001	10
–0.39	1 0	11	01	01
–0.12	1 0	11	01	00
0.12	0 0	00	10	00
0.39	0 0	00	10	01
0.69	1 10	101	110	10
1.06	1 10	101	110	110
1.51	0 110	0110	1110	1110
2.14	0 110	0110	1110	11110
3.09	1 110	1000	1111	111110
5.12	1 110	1000	1111	111111

2) Index each branch from the left (right) side of the symmetric tree by flipping the index of its image from the right (left) side.

Table I shows three different possible index assignments for an eight-point scalar quantizer with a specific set of codeword lengths. Code(I) and Code(II) are the two choices obtained by the abovementioned procedure and Code(III) is not satisfying the rules.

Table II shows different index assignments for a 16-points scalar quantizer to be used in our FE-TCQ with two different code-word length sets [23]. Note that each index is repeated two times due to the natural redundancy in the trellis structure of the underlying TCQ. In Code(I), a prefix code with code words $\{111, 110, 10, 0\}$ is designed for each subset $S_i, i \in \{0, 1, 2, 3\}$. A different bit is added as an index to differentiate between S_0 and S_2 or S_1 and S_3 (there are four elements in each subset and the index assignment presented before does not have any impact on the labeling). In all sets of Code(II), a prefix code is designed for $S_0 \cup S_2$ and $S_1 \cup S_3$. Thus, no bit is required as an indicator for each subset. Code(II,a) and (II,b) are designed for a specific set of lengths, which is different from Code(II,c). Therefore, the differences between (II,a) and (II,b) with Code(IIc) is the matter of a different prefix code rather than a different index assignment. In designing Code(II,a), the index assignment rules are not satisfied while they are in Code(II,b).

III. TAIL-BITING TRELLIS-CODED QUANTIZATION (TB-TCQ)

Consider an N-D TCQ (N is the block length) with $\nu = 2^{\mu}$ states. The bit sequence at the quantizer output specifies the choice among the possible branches at each state plus an additional μ bits, which specify the starting state. These additional μ bits will have a strong impact on the effective bit rate for small values of block length.

Two modifications can be made to avoid sending the extra μ bits. First, one can use a fixed starting state TCQ to encode each block of source samples. As we will see later, one will loose part of the achievable granular gain by imposing this constraint.

As an alternative, we propose to use a tail-biting trellis structure in which the start and end states on the trellis paths are the same. Using this structure, one does not need any extra bits to specify the starting state. The idea is similar to the tail-biting trellis structure used to construct a block code from a convolutional code [27]–[29]. To search through a tail-biting trellis, one straightforward procedure is to run the Viterbi algorithm ν times for different starting states. The main disadvantage of this method is the increase in the computational complexity (since the Viterbi algorithm has to be used ν times, the encoding computational complexity of the tail-biting TCQ is ν times higher than that of TCQ [24]). To reduce the complexity, several suboptimum and optimum search algorithms have been proposed in the context of the convolutional codes [27], [28].

In order to overcome the complexity issue of the tail-biting trellis structure, we present a possible suboptimum search method and a new trellis structure. The new structure, which we call "modified tail-biting trellis" (MTB-TCQ), consists of all paths with a fixed starting and ending state. The encoding complexity of the modified tail-biting trellis is the same as that of the conventional TCQ. Although this structure may loose some part of the packing gain, it can achieve more boundary gain when it is combined with an FEVQ due to using less bits to represent the trellis path. (Note that less paths exist in the modified trellis.) This gain is noticeable when the FEVQ has not achieved most of the boundary gain by itself. We explain this issue in more details when we present our numerical results.

The main idea for applying a suboptimum search method is to reduce the number of independent TCQ search iterations. In the following, we present a suboptimum search algorithm that requires just one search iteration.

Algorithm

1) Choose all states as the starting state.
2) At stage $N - \mu$, check the winning path. Extract the starting state of the winning path (S_i).
3) At stage $N - \mu$, set the metric of all the path that does not start from state S_i to infinity (forcing all states at stage $N - \mu$ to end at state S_i at stage N).
4) Continue the encoding until the last stage.
5) Path starting from S_i and ending to S_i is the final answer.

IV. NUMERICAL RESULTS

In this section, we present numerical results for the performance of the proposed methods. In all cases, a sequence of 600 000 source samples is used to design the quantizer (using an iterative design procedure [30]) and a different sequence of the same length is used to measure the corresponding performance. Input samples are from memoryless Gaussian or Laplacian sources. The underlying scalar quantizer is composed of

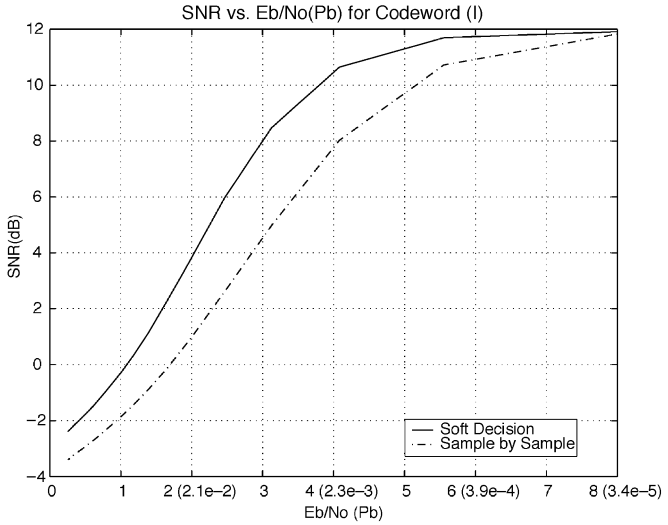

Fig. 4. Comparison of received signal-to-noise ratio (SNR) for FEVQ using ML decoder and sample-by-sample decoder for different $E_b/N_0(P_b)$ [Code(I) of Table I is used], Gaussian source.

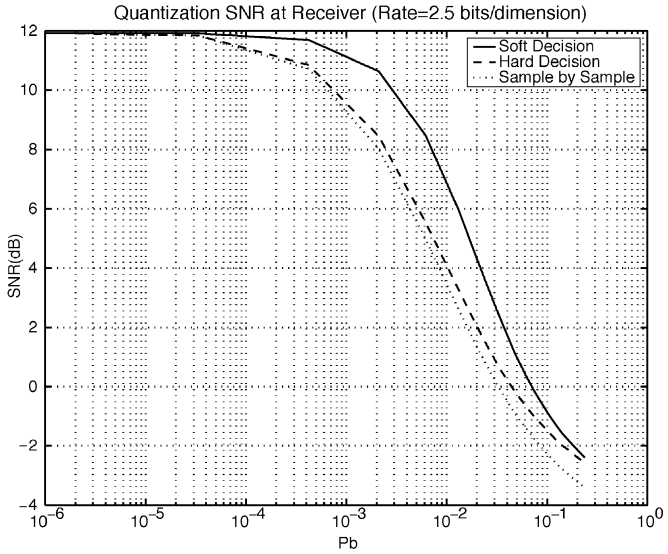

Fig. 5. Comparison of received SNR for soft, hard-decision decoding using trellis structure and sample-by-sample decoding for different bit error probabilities of BPSK channel P_b [Code(I) of Table I is used] Gaussian source.

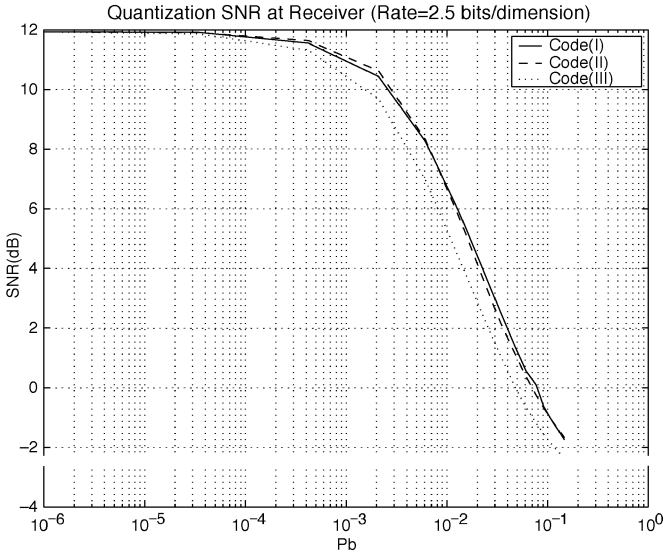

Fig. 6. Comparison of soft-decision decoding for different binary codes of Table I as a function of bit error probability of BPSK channel, P_b, Gaussian source.

eight points and, for the case with TCQ structure, the number is 16. Space dimension is $N = 32$ and $L_{\max} = 80$, resulting in an effective rate of 2.5 b/sample (for FEVQ and FE-TCQ). The quantization performance is measured in terms of the mean square distance. The channel model is AWGN with BPSK modulation.

First, we present the numerical results of the proposed ML decoder for FEVQ and FE-TCQ. Fig. 4 shows the received SNR versus channel E_b/N_0 for ML decoding and for the sample-by-sample decoding of FEVQ.

In many known source-channel coding schemes, unlike the method discussed in the paper, the encoder and/or decoder are optimized to operate over a specific channel. Therefore, a comparison between the proposed method and this type of joint source-channel coding schemes is not appropriate. However, another class of source-channel decoders, namely "zero redundancy source-channel decoder," take advantage of the residual redundancy existing at the encoder output to protect the data against the channel noise. Since our FEVQ is based on variable-length codes (prefix codes), we provide comparisons with the source-channel decoder presented in [13], which uses variable-length codes and takes advantage of the residual redundancy in the decoding. In [13], at a BER of 10^{-2}, there has been an improvement of 0.3 dB over the case where the residual redundancy is not exploited. Fig. 4 shows that, for the proposed ML decoder, the transmitter power can be reduced between one and two decibels. For example, at SNR $= 4$ dB, the difference in the required E_b/N_0 for the two systems is approximately 1.4 dB. This comparison shows the superior performance of our method.

Fig. 5 shows the comparison between three different decoding methods, ML decoding approach using hard and soft decision, and sample-by-sample decoding. The structure of the code words is the same as Code(I) in Table I. It is observed that the ML decoding using soft-decision results in an approximately 1–3-dB improvement in comparison with the other two alternatives. Fig. 6 shows a comparison of soft-decision decoding for prefix code with three different labeling methods given in Table I. As expected, Code(I) and Code(II) show almost the same performance, while they are both superior in performance in comparison with Code(III).

Assume the prefix code of FEVQ has the set of $\{\ell_1, \ell_2, \ldots, \ell_M\}$ as the code-word length $\ell_{\min}$ as the minimum codeword length and M' as the distinct number of code-word lengths. The total allowed path or the total states at stage N (refer to Fig. 1) will be $L_{\max} - N\ell_{\min}$. To keep the information of these paths in all the stages, a total of $(L_{\max} - N\ell_{\min})N$ units of memory is required. We also require $2L_{\max}$ units of memory to keep the Euclidean distance of every received packet of $L_{\max}$ bits to BPSK level of 1 and -1. There is no multiplication required to calculate the Euclidean distance [the two terms of $(x-1)^2 = (x^2+1) - 2x$ and $(x+1)^2 = (x^2+1) + 2x$

TABLE III
COMPLEXITY OF FEVQ AT A RATE OF 2.5 B/DIMENSION. MEMORY SIZE IS IN BYTE PER N DIMENSIONS AND THE NUMBER OF ADDITIONS/MULTIPLICATIONS ARE COUNTED PER DIMENSION

	Addition	Memory
FEVQ	510	0.7 k
FE-TCQ (Code(I))	280	0.6 k
FE-TCQ (Code(II,a))	2040	2.4 k

TABLE IV
SOFT DECODING OF FE-TCQ USING CODE(I) AND (II,C) OF TABLE II FOR DIFFERENT BIT ERROR PROBABILITIES FOR A FOUR-STATE TRELLIS AND $2M = 16$ POINTS SCALAR QUANTIZER, $N = 32$, $R = 2.5$ b/SAMPLE, GAUSSIAN SOURCE

	$P_b = 0.01$	$P_b = 0.005$	$P_b = 0.001$	$P_b = 0$
SNR, Code I	6.51	8.54	11.24	12.61
SNR Code II,c	1.65	3.96	8.91	13.35

can be substituted by $-x$ and x, respectively, without affecting the decisions]. Since there are M branches leaving each state of the trellis in Fig. 1, there are $\sum_{i=1}^{M}(\ell_i - 1)$ additions required to calculate the metric of the M branches. There are also M' additions required to find the total metric of the M' paths ending to the next stage. Therefore, the total number of additions will be $\sum_{i=1}^{M}(\ell_i - 1) + M'$ times the total number of states in the trellis.

FE-TCQ presents a different complexity based on the way prefix coding is done for the subsets (designing a prefix code for each subset or for the union of pair of them). For Type(I), the TCQ decoder and ML decoder of FEVQ can be separated into two modules that can work in parallel. Therefore, the complexity of the FE-TCQ at rate R will be the sum of the TCQ complexity at rate 1 and ML decoder complexity at rate $R-1$. For Type(II), the TCQ decoder and ML decoder of FEVQ act as one module and they cannot be separated any more. In this case, the total allowed paths or the total number of the states at stage N will be $(L_{\max} - N\ell_{\min})\mu$, where μ is the number of the states in TCQ trellis. To keep the information of these paths at all stages, a total of $(L_{\max} - N\ell_{\min})N\mu$ units of memory is required. The number of multiplications remains zero, the same as that of the FEVQ, and the number of additions is multiplied with the number of TCQ states (μ).

Table III presents the computational complexity of an ML decoder for FEVQ and FE-TCQ at the rate of 2.5 b/dimension. As anticipated, the FE-TCQ using Code(II) has higher complexity than Code(I). It is shown that the achieved gain due to the ML decoder is at the cost of a negligible complexity.

Table IV shows the comparison of the soft-decision decoding of the FE-TCQ over a noisy channel for Code(I) and Code(II,c) from Table II. Although Code(II,c) shows better performance in an error-free channel, Code(I) is more robust in the presence of channel noise. This is because the prefix code with the least expected length is more sensitive to the channel noise. This means that the synchronization happens faster in a code with a larger expected length. Another way of explaining the results is that the less the redundancy, the less immunity from the channel noise.

Fig. 7 compares the performance of the FE-TCQ using four different index assignments given in Table II. Because of the proper index assignment, Code(II,b) shows better performance than Code(I) and Code(II,a). Code(II,c) has worse performance

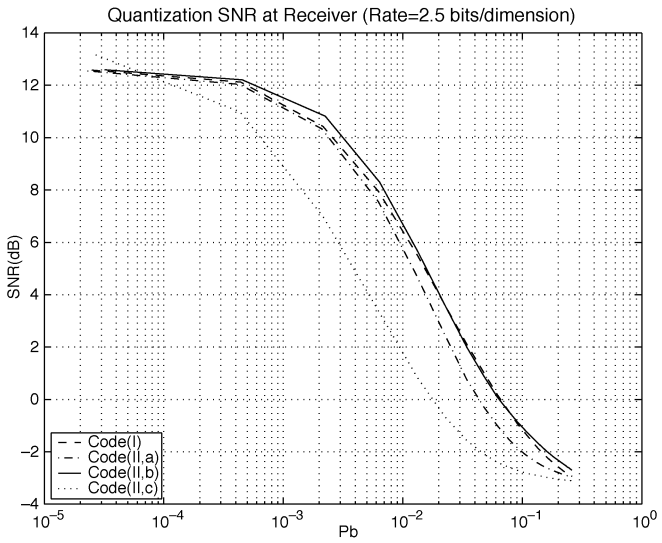

Fig. 7. Comparison of soft-decision decoding for different binary codes of Table II as a function of bit error probability of BPSK channel P_b, Gaussian source.

in a noisy channel (against its best performance in noiseless channel) due to less redundancy in comparison with Code(I), Code(II,a), and Code(II,b).

We have included some numerical results for the methods proposed in [25] and [26]. The results correspond to the performance of modified trellis-based scalar-vector quantizer (TB-SVQ) in a noisy channel (refer to Table IX). This is a fixed-rate trellis-coded quantizer that uses an enumerating algorithm to label the code-word sequences. By comparison with Table IV, TB-SVQ outperforms FE-TCQ in a noiseless channel. However, in a noisy channel, the proposed ML decoder substantially improves the FE-TCQ performance, resulting in a better performance than TB-SVQ.

In the second set of numerical results, we compare the performance of the tail-biting trellis structure with other fixed block length trellises, as well as TCQ and TCQ, with an early decision. Tables V and VI provide a comparison between TB-TCQ and TCQ for both Gaussian and Laplacian data. We present numerical results for three cases of the TCQ. In the first, TCQ has a block length of 1000 samples[2] and, in the second case, the starting state is fixed, which we call the fixed-starting-state TCQ (FS-TCQ). In the third case, $N = 600\,000$, block length is used with an early decision after $N = 16$ stages (decoding depth of the Viterbi algorithm is 16). It is observed that TB-TCQ has a performance similar to TCQ with a much smaller delay. It is also observed that TB-TCQ offers a slight improvement over truncated TCQ. Tables VII and VIII compare the performance of the full-search algorithm with that of the suboptimum search algorithm. The results show a small gap between their performances. For the Gaussian data, this gap becomes smaller as the number of states increases, but for Laplacian data, as the number of states increases, the gap becomes larger. Fig. 8 show the end-to-end performance of the TB-TCQ and TCQ $(N = 1000)$

[2]This is the same block length as used in [24]. It should be mentioned that the numerical results presented for $N = 1000$ are computed independently of [24] (following the same setup as used for all our numerical results) and are slightly different from the values reported in [24].

TABLE V
QUANTIZATION SNR (IN DECIBELS) FOR THE FOUR, EIGHT, 16, AND 32 STATES TRELLIS DIAGRAMS, $R = 2$ b/SAMPLE, GAUSSIAN SOURCE

	4 State		8 State		16 State		32 State	
Block Length	TB-TCQ	FS-TCQ	TB-TCQ	FS-TCQ	TB-TCQ	FS-TCQ	TB-TCQ	FS-TCQ
16	10.47	10.16	10.52	10.18	10.53	10.11	10.52	9.96
20	10.50	10.23	10.58	10.27	10.60	10.21	10.61	10.08
24	10.52	10.28	10.62	10.32	10.64	10.29	10.68	10.19
28	10.52	10.32	10.63	10.38	10.67	10.34	10.71	10.26
32	10.53	10.34	10.65	10.41	10.70	10.39	10.74	10.33
64	10.53	10.43	10.66	10.51	10.74	10.56	10.80	10.56
128	10.53	10.48	10.66	10.59	10.74	10.64	10.80	10.68
256	10.53	10.51	10.66	10.62	10.74	10.69	10.80	10.74
512	10.53	10.53	10.66	10.64	10.74	10.71	10.80	10.77
1024	10.53	10.53	10.66	10.65	10.74	10.72	10.80	10.79
TCQ ($N = 1000$, [24])	10.54		10.67		10.75		10.82	
TCQ (Truncated,$N = 16$)	10.42		10.44		10.46		10.44	

TABLE VI
QUANTIZATION SNR (IN DECIBELS) FOR THE FOUR, EIGHT, 16, AND 32 STATES TRELLIS DIAGRAMS $R = 2$ b/SAMPLE, LAPLACIAN SOURCE

	4 State		8 State		16 State		32 State	
Block Length	TB-TCQ	FS-TCQ	TB-TCQ	FS-TCQ	TB-TCQ	FS-TCQ	TB-TCQ	FS-TCQ
16	9.38	9.02	9.48	9.06	9.49	8.98	9.50	8.72
20	9.39	9.09	9.51	9.15	9.56	9.09	9.58	8.88
24	9.39	9.15	9.53	9.22	9.60	9.17	9.62	8.99
28	9.40	9.19	9.54	9.28	9.62	9.24	9.65	9.10
32	9.41	9.21	9.54	9.31	9.63	9.29	9.67	9.14
64	9.41	9.31	9.55	9.44	9.65	9.48	9.71	9.41
128	9.41	9.37	9.56	9.51	9.65	9.56	9.71	9.54
256	9.41	9.40	9.56	9.54	9.65	9.61	9.71	9.62
512	9.41	9.41	9.56	9.56	9.65	9.63	9.71	9.65
1024	9.41	9.41	9.56	9.57	9.65	9.64	9.71	9.68
TCQ ($N = 1000$ [24])	9.41		9.56		9.67		9.73	
TCQ (Truncated, $N = 16$)	9.34		9.40		9.44		9.43	

TABLE VII
QUANTIZATION SNR (IN DECIBELS) FOR FULL SEARCH TB-TCQ AND THE SUBOPTIMUM SEARCH TB-TCQ FOR DIFFERENT TRELLIS STRUCTURES $N = 128$, $R = 2$b/SAMPLE, GAUSSIAN SOURCE

	4-state	8-state	16-state	32-state
Full search	10.53	10.64	10.70	10.74
Sub-optimum search	10.50	10.60	10.65	10.70

TABLE VIII
QUANTIZATION SNR (IN DECIBELS) FOR FULL SEARCH TB-TCQ AND THE SUBOPTIMUM SEARCH TB-TCQ FOR DIFFERENT TRELLIS STRUCTURES $N = 128$, $R = 2$ b/SAMPLE, LAPLACIAN SOURCE

	4-state	8-state	16-state	32-state
Full search	9.40	9.56	9.65	9.71
Sub-optimum search	9.37	9.48	9.56	9.59

TABLE IX
PERFORMANCE OF THE FIXED-RATE TCQ SCHEME PROPOSED IN [26], USING A FOUR-STATE TRELLIS, GAUSSIAN SOURCE

Bit rate	$P_b = 0.01$	$P_b = 0.005$	$P_b = 0.001$	$P_b = 0$
2	1.53	3.69	8.19	11.15
3	0.5	2.74	8.61	16.71

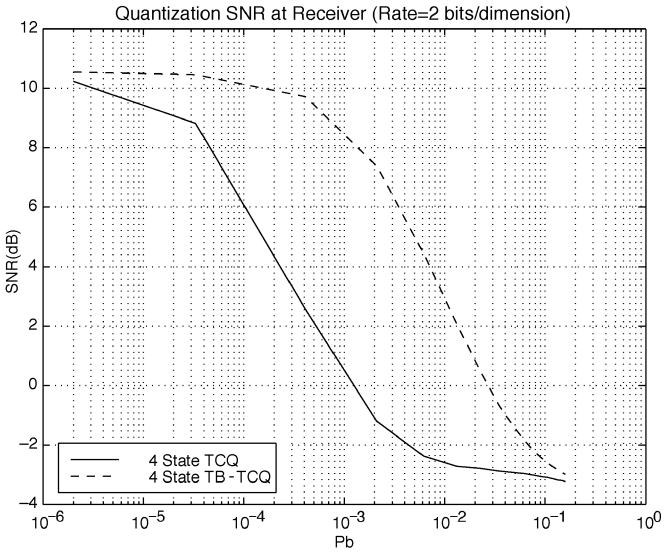

Fig. 8. Comparison of the end-to-end quantization SNR (in decibels) for four-state TCQ ($N = 1000$) and TB-TCQ ($N = 32$) for different probabilities of bit error (P_b), Gaussian source.

as a function of the resulting bit error probability. These curves demonstrate a significant improvement for TB-TCQ in comparison with TCQ. Similar improvements are observed for trellis diagrams with a larger number of states.

Tables X and XI present the comparison of FE-TCQ for the different trellis structures. These comparisons clarify more than the advantage of the tail-biting trellis structure over the other methods for trellis termination. All the results correspond to $R = 2.5$ b/sample (the total binary length of FEVQ is $L_{\mathrm{Max}} = 80$). In some cases, a few more bits are required to specify the starting state, depending on the type of trellis structure (in which case the total binary length will be modified accordingly). For the regular trellis diagram (TCQ), the numbers of those bits are two or three (for four or eight state trellis diagrams, respectively) and there are no bits required to specify the starting state for the FS-TCQ and TB-TCQ trellis structures. For the MTB-TCQ

TABLE X
QUANTIZATION SNR (IN DECIBELS) FOR FE-TCQ FOR DIFFERENT TRELLIS STRUCTURES. TRELLISES HAVE FOUR AND EIGHT STATES. THERE ARE 16 THRESHOLD POINTS ALONG EACH DIMENSION, $R = 2.5$ b/SAMPLE, GAUSSIAN SOURCE

	4-state trellis			
	TCQ	FS-TCQ	TB-TCQ	MTB-TCQ
Code(I)	12.61	12.78	12.94	12.97
Code(II,c)	13.35	13.31	13.45	13.27
	8-state trellis			
	TCQ	FS-TCQ	TB-TCQ	MTB-TCQ
Code(I)	12.55	12.83	13.04	13.10
Code(II,c)	13.39	13.33	13.50	13.41

TABLE XI
QUANTIZATION SNR (IN DECIBELS) FOR FE-TCQ FOR DIFFERENT TRELLIS STRUCTURES. TRELLISES HAVE FOUR AND EIGHT STATES. THERE ARE 16 THRESHOLD POINTS ALONG EACH DIMENSION, $R = 2.5$ b/SAMPLE, LAPLACIAN SOURCE

	4-state trellis			
	TCQ	FS-TCQ	TB-TCQ	MTB-TCQ
Code(I)	12.8	12.89	13.07	12.90
Code(II,c)	13.23	13.13	13.33	13.13
	8-state trellis			
	TCQ	FS-TCQ	TB-TCQ	MTB-TCQ
Code(I)	12.75	12.90	13.16	12.95
Code(II,c)	13.31	13.15	13.45	13.05

TABLE XII
RECEIVED SNR (IN DECIBELS) FOR SOFT DECODING OF FIXED-RATE ENTROPY-CODED QUANTIZER USING DIFFERENT TRELLIS STRUCTURES FOR DIFFERENT BIT PROBABILITY OF ERROR (P_b). THE CODE(II,B) IS USED, $R = 2.5$ b/SAMPLE, GAUSSIAN SOURCE

	$P_b = 0.01$	$P_b = 0.005$	$P_b = 0.001$	$P_b = 0$
TCQ	6.67	9.00	11.66	12.61
TB-TCQ	8.14	10.65	12.20	12.94
FS-TCQ	6.82	8.98	11.87	12.80
MTB-TCQ	7.97	10.51	12.78	12.97

trellis structure, the encoder not only does not need any extra bits for the starting state, but it also saves another two or three bits. These saved bits correspond to the last two or three stages of the trellis as the trellis path must end at a fixed state. (Since the ending state is a fixed state, there is only one path from the stages $N-2$ or $N-3$ to the stage N.) In other words, since the starting and ending states for this quantizer are the same, the decoder does not require the transmission of the last two or three path indicator bits. In summary, if the trellis structure has $\nu = 2^{\mu}$ states, the actual total binary length for the regular trellis is $L_{\max} - \mu$; for the modified tail biting, it is $L_{\max} + \mu$ and for the tail-biting and fixed starting state, it is $L_{\max}$. Therefore, if FEVQ has not already achieved most of the boundary gain, the FE-TCQ using the modified tail-biting trellis may show better performance than when a tail-biting trellis is used. As an example for Gaussian data, the FE-TCQ with Code(I) and modified tail-biting trellis performs slightly better than the case with a tail-biting trellis (refer to Table X).

The numerical results of this comparison for the four-state trellis structure and two different prefix code assignments indicate that prefix Code(II,c) produces better results for all cases. The same trend is also observed for the eight-state trellis, except for the fixed-starting state case. The same results is observed in Tables V and VI; the fixed-starting state trellis performs even worse when the number of states increases.

TABLE XIII
RECEIVED SNR (IN DECIBELS) FOR SOFT DECODING OF FIXED-RATE ENTROPY-CODED QUANTIZER USING DIFFERENT TRELLIS STRUCTURES FOR DIFFERENT BIT PROBABILITY OF ERROR (P_b). THE CODE(II,C) IS USED, $R = 2.5$ b/SAMPLE, GAUSSIAN SOURCE

	$P_b = 0.01$	$P_b = 0.005$	$P_b = 0.001$	$P_b = 0$
TCQ	1.75	4.03	9.04	13.36
TB-TCQ	2.71	5.56	10.49	13.46
FS-TCQ	1.86	4.21	9.34	13.31
MTB-TCQ	3.08	5.78	10.57	13.27

Tables XII and XIII show the comparison of soft-decision decoding of the FE-TCQ using the four different trellis structures. The prefix Code(II,c) and Code(II,b) are chosen from Table II for this comparison. In summary, the fixed-rate entropy-coded quantizer using a tail-biting trellis structure shows a better performance in comparison with other trellis structures. Prefix Code(II,c) has the best performance in an error-free channel, whereas prefix Code(II,b) shows the most robust performance in a noisy channel.

REFERENCES

[1] S. Nikneshan and A. K. Khandani, "Soft decision decoding of fixed rate entropy constrained quantizer over a noisy channel," in *Proc. 20th Biennial Symp. Commun.*, Kingston, ON, Canada, May 2000, pp. 116–118.

[2] T. J. Ferguson and J. J. Rabinowitz, "Self-synchronizing Huffman codes," *IEEE Trans. Inform. Theory*, vol. IT-30, pp. 687–693, July 1984.

[3] B. L. Montgomery and J. Abrahams, "Synchronization of binary source codes," *IEEE Trans. Inform. Theory*, vol. IT-32, pp. 849–854, Nov. 1986.

[4] L. R. Bahl, J. Cocke, F. Jelinek, and J. Raviv, "Optimal decoding of linear codes for minimizing symbol error rate," *IEEE Trans. Inform. Theory*, vol. IT-20, pp. 284–287, Mar. 1974.

[5] N. Demir and K. Sayood, "Joint source-channel decoding for variable-length codes," in *Proc. Data Comp. Conf.*, Snowbird, UT, Mar. 1998, pp. 139–148.

[6] ——, "Joint source-channel decoding for variable-length encoded data by exact and approximate MAP sequence estimation," *IEEE Trans. Commun.*, vol. 48, pp. 1–6, Jan. 2000.

[7] K. P. Subbalakshmi and J. Vaisey, "Optimal decoding of entropy coded memoryless sources over binary symmetric channel," in *Proc. Data Comp. Conf.*, Snowbird, UT, Mar. 1998, p. 573.

[8] A. H. Murad and T. E. Fuja, "Joint source-channel decoding of variable-length encoded sources," in *Proc. Information Theory Workshop*, Killarney, Ireland, June 1998, pp. 94–95.

[9] M. Park and D. Miller, "Improved joint source-channel decoding for variable-length encoded data by using soft decisions and MMSE estimation," in *Proc. Data Comp. Conf.*, Snowbird, UT, 1999, pp. 29–31.

[10] K. Sayood and J. C. Borkenhagen, "Use of residual redundancy in the design of joint source/channel coders," *IEEE Trans. Commun.*, vol. 39, pp. 838–846, June 1991.

[11] R. Bauer and J. Hagenauer, "Iterative source/channel-decoding using reversible variable-length codes," in *Proc. Data Comp. Conf.*, Snowbird, UT, 2000, pp. 93–102.

[12] A. Guyaderi, E. Fabre, C. Guillemot, and M. Robert, "Joint source-channel turbo decoding of entropy-coded sources," *IEEE J. Select. Areas Commun.*, vol. 19, pp. 1680–1696, Sept. 2001.

[13] M. Jeanne, J. C. Carlach, P. Sihon, and L. Guivarch, "Source and joint source-channel decoding of variable length codes," in *Proc. IEEE Int. Conf. Commun. (ICC'02)*, vol. 2, 2002, pp. 768–772.

[14] R. Laroia and N. Farvardin, "A structured fixed-rate vector quantizer derived from variable-length scalar quantizer—Part I: Memoryless sources," *IEEE Trans. Inform. Theory*, vol. 39, pp. 851–867, May 1993.

[15] A. K. Khandani, "A hierarchical dynamic programming approach to fixed-rate, entropy-coded quantization," *IEEE Trans. Inform. Theory*, vol. 42, pp. 1298–1303, July 1996.
[16] ——, "A linear (zero-one) programming approach to fixed-rate entropy-coded vector quantization," *Proc. Inst. Elect. Eng. Commun.*, vol. 146, no. 5, pp. 275–282, 1999.
[17] K. A. Zeger and A. Gersho, "Zero redundancy channel coding in vector quantization," *IEEE Trans. Inform. Theory*, vol. IT-33, pp. 654–655, June 1987.
[18] N. Farvardin and V. Vaishampayan, "Optimal quantizer design for noisy channel: An approach to combined source-channel coding," *IEEE Trans. Inform. Theory*, vol. 33, pp. 827–837, Nov. 1987.
[19] S. Nikneshan, "Lattice/Trellis Based Fixed-Rate Entropy-Coded Vector Quantization," Ph.D. dissertation, Elect. Comput. Eng. Dept., Univ. of Waterloo, Waterloo, ON, Canada, Oct. 2001.
[20] J. M. Wozencraft and I. M. Jacobs, *Principles of Communication Engineering*. New York: Wiley, 1965.
[21] T. M. Cover and J. A. Thomas, *Elements of Information Theory*. New York: Wiley, 1991.
[22] T. R. Fischer and M. Wang, "Entropy-constrained trellis-coded quantization," *IEEE Trans. Inform. Theory*, vol. 38, pp. 415–426, Mar. 1992.
[23] M. W. Marcellin, "On entropy constrained TCQ," *IEEE Trans. Commun.*, vol. 42, pp. 14–16, Jan. 1994.
[24] M. W. Marcellin and T. R. Fischer, "Trellis-coded quantization of memoryless and Gauss–Markov sources," *IEEE Trans. Commun.*, vol. 38, pp. 82–93, Nov. 1990.
[25] R. Laroia and N. Farvardin, "Trellis-based scalar-vector quantizer for memoryless sources," *IEEE Trans. Inform. Theory*, vol. 40, pp. 860–870, May 1994.
[26] L. Yang and T. R. Fischer, "A new trellis source code for memoryless sources," *IEEE Trans. Inform. Theory*, vol. 44, pp. 3056–3063, Nov. 1998.
[27] H. H. Ma and J. K. Wolf, "On tail biting convolutional codes," *IEEE Trans. Commun.*, vol. COM-34, pp. 104–111, Feb. 1986.
[28] Q. Wang and V. K. Bhargava, "An efficient maximum likelihood decoding algorithm for generalized tail biting convolutional codes including quasicyclic codes," *IEEE Trans. Commun.*, vol. 37, pp. 875–879, Aug. 1989.
[29] G. Solomon and H. C. A. van Tilborg, "A connection between block and convolutional codes," *SIAM J. Appl. Math.*, vol. 37, pp. 358–369, Aug. 1989.
[30] Y. Linde, A. Buzo, and R. M. Gray, "An algorithm for vector quantization design," *IEEE Trans. Commun.*, vol. COM-29, pp. 84–95, Jan. 1980.

Sasan Nikneshan (M'04) received the B.S. and M.S. degrees from the University of Tehran, Tehran, Iran, in 1992 and 1995, respectively, and the Ph.D. degree from the University of Waterloo, Waterloo, ON, Canada, in 2001, all in electrical engineering.

After graduation, he joined Valence Semiconductor, Inc., Toronto, Canada, where he was the System Architect for the digital visual interface (DVI) group. Currently, he is with RFTune, Inc., Waterloo, where he is working on a smart antenna transceiver system and architecture for a wireless local area network (WLAN; IEEE 802.11) application.

Dr. Nikneshan was the recipient of the Ontario Graduate Scholarship from 1998 to 1999 and the Ontario Graduate Scholarship in Science and Technology from 1999 to 2000, as well as five Faculty of Engineering Awards at the University of Waterloo.

Amir K. Khandani (M'02) received the M.Eng. degree from the University of Tehran, Tehran, Iran, and the Ph.D. degree from McGill University, Montreal, QC, Canada, in 1985 and 1992, respectively.

Following that, he was a Research Associate at the INRS-Telecommunication, Montreal. In 1993, he joined the Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, ON, Canada, where he is currently a Professor and an Industrial Research Chair Holder, funded by Nortel Networks.

He is currently serving as an Associate Editor for IEEE TRANSACTIONS ON COMMUNICATIONS in the area of coding and communication theory.

Peter Kabal (S'70–M'75) received the Ph.D. degree in electrical engineering from the University of Toronto, Toronto, ON, Canada, in 1975.

He is a Professor of electrical and computer engineering at McGill University, Montreal, QC, Canada, and holds the Natural Science and Engineering Research Council of Canada/Nortel Industrial Research Chair. His current research interests focus on digital signal processing (DSP) as applied to speech and audio processing, adaptive filtering, and data transmission.