Design of Successively Refinable Trellis-Coded Quantizers

Hamid Jafarkhani, *Member, IEEE*, and Vahid Tarokh, *Member, IEEE*

***Abstract*—We propose successively refinable trellis-coded quantizers for progressive transmission. (Progressive transmission is an essential component of image and multimedia browsing systems.) A new trellis structure which is scalable is used in the design of our trellis-coded quantizers. A hierarchical set partitioning is developed to preserve successive refinability. Two algorithms for designing trellis-coded quantizers which provide embedded bit streams are provided. The computational complexity of the proposed schemes is compared with that of trellis-coded quantization. Simulation results show excellent performances for memoryless sources.**

***Index Terms*—Embedded bit stream, image browsing, progressive transmission, successive refinability, trellis-coded quantization.**

I. Introduction

In recent years, successively refinable or rate-scalable source coders have received growing attention. An embedded output bit stream is provided by successively refinable source coders, i.e., by selecting different substreams of the output, various levels of encoding rate and distortion can be achieved.

One immediate application of rate-scalability is in progressive transmission. Sometimes the utility of progressive transmission is readily apparent because it is central to an application. One important example is image browsing (for example, over the World Wide Web). In image browsing, one may need to access a large image over a slow network connection. Such a transmission requires a large delay which is not desirable. Instead of waiting for the entire bit stream to reconstruct the image, successive refinability provides the opportunity of progressive transmission. So, the decoder can provide a higher quality replica of the image based on receiving each new packet. Another application is the transmission of maps in which a low-resolution, low-quality map of a large area may be first transmitted. The user may then identify a specific area in the received map and request a high-resolution, high-quality map of the specified region. A similar application is in telemedicine, where a specialist must sort through and retrieve a large number of medical images over a network. If the images can be transmitted progressively so that unneeded images or sections of images can be identified before they have been transmitted with high fidelity, then both cost and bandwidth may be saved.

Another application of an embedded bit stream is in the transmission of video over an asynchronous transfer mode (ATM) network where some ATM cells may be lost in transit through the network. A video coder with an embedded output bit stream can mark the ATM cells with an importance measure; as a video frame is coded progressively the descriptions become successively less important since they contribute less to the reduction of distortion. Cells in the network can then be dropped preferentially based upon their importance in distortion reduction. Also, when multicasting over heterogeneous networks, rate-scalability provides the opportunity of using one bit stream for all receivers and intelligently dropping the less important portions of the bit stream for users with less available bandwidth.

Additionally, in many practical applications where the signal is transmitted over a noisy channel, the rates of the source and channel codes must be adjusted according to the level of noise in the channel. If the channel is a time-varying channel, as in many wireless communication situations, it is prudent to adaptively vary the rate allocation between the source and channel coding operations. Successive refinability allows the possibility of adapting the rate of the source encoder in a straightforward manner; of course, similar rate-scalability features are needed for the channel coding part [1]–[3].

Information-theoretic results concerning successively refinable encoders have been reported in the literature [4]–[8]. These results indicate that, asymptotically in block length and for some source distributions, there exist successively refinable vector quantizers that achieve the rate-distortion bound. Source distributions for which such successively refinable quantizers exist are called *successively refinable* [7]. Little is known about the performance of successively refinable quantizers with finite block length, either for successively refinable distributions or nonsuccessively refinable distributions.

An example of an important implementation of a successively refinable quantizer for speech coding is given in [9]. The design of fixed-rate and entropy constrained successively refinable scalar quantizers are considered in [10]. Since it is usually impossible to simultaneously optimize the rate-distortion performance in all stages, a weighted average of the distortion at each stage of refinement has been minimized as a design criterion in [10]. A different approach to solve a related problem can be found in [11].

Manuscript received December 1, 1997; revised November 2, 1998. The material in this paper was presented in part at the IEEE Data Compression Conference, Snowbird, UT, March 1998.

H. Jafarkhani is with AT&T Labs–Research, Red Bank, NJ 07701 USA (e-mail: hamid@research.att.com).

V. Tarokh is with AT&T Shannon Laboratories, Florham Park, NJ 07932 USA (e-mail: tarokh@research.att.com).

Communicated by R. Laroia, Associate Editor for Source Coding.

Publisher Item Identifier S 0018-9448(99)04357-6.

A powerful source coding scheme for memoryless sources is trellis-coded quantization [12]. It has been shown that for a memoryless uniform source, trellis-coded quantizers (TCQ's) provide mean-squared errors (MSE's) within 0.21 dB of the theoretical distortion bounds (for given rates) [12]. The performance of a TCQ is much better than that of the best scalar quantizer (Lloyd–Max quantizer) at the same rate. TCQ has been used in many speech and image coding systems. A recent image coding system based on wavelet transform, classification, and entropy constrained TCQ provides one of the best available rate-distortion performances in the literature [13].

Unfortunately, the TCQ proposed in [12] does not lend itself to successive refinability. Previous attempts in designing a TCQ which can be used for progressive transmission include the schemes presented in [14] and [15]. In [14], the idea of multistage quantization is combined with that of TCQ. In other words, each stage of TCQ quantizes the error between the original and the quantized output of the previous stage. Unfortunately, multistage TCQ (MS-TCQ) suffers from about 2-dB performance degradation compared to the performance of TCQ. In [15], TCQ structure of [12] is preserved and trellis coding is only applied to the last stage. Using a successive approximation-type setting, the inverse TCQ operation is performed only approximately for all intermediate stages. The resulting quantization scheme has been utilized for progressive transmission of images.

In this work, we present a new structure for TCQ's which is inherently rate-scalable. We also propose two new algorithms to design rate-scalable TCQ's and report their performance for memoryless sources. The organization of the paper is as follows: Section II presents the new trellis structure for TCQ. In Section III, we introduce trellis-coded quantizers which are successively refinable. Section IV proposes two design algorithms and Section V analyzes the computational complexity of the proposed quantizers. Simulation results and concluding remarks are provided in Section VI.

II. A Tree-Structured Trellis for Successive Refinability

The TCQ of [12] is based on Ungerboeck's trellis structures and set partitioning principles [16]. Fig. 1 shows Ungerboeck's four-state amplitude modulation trellis which is also used in [12]. Assume that it is desired to quantize and transmit a stationary and ergodic random process $X = \{X(n)\}_{n=1}^{\infty}$ with marginal probability density function (pdf) $p_X(x)$, $x \in \mathbb{R}$. To transmit R bits per sample, 2^{R+1} codevectors[1] in $\mathbb{R}$ (doubled codebook) are partitioned into four subsets, D_0, D_1, D_2, and D_3. (In general, the codevectors can be partitioned into $2^{\tilde{m}+1}$ subsets. In this work, we only consider $\tilde{m} = 1$ for the sake of brevity.) Then the best output sequence, $\hat{X} = \{\hat{X}(n)\}_{n=1}^{\infty}$, is the one which minimizes the per-symbol MSE defined by $\mathcal{D} = E[(X(n) - \hat{X}(n))^2]$. Such a sequence is achieved by using the Viterbi algorithm [17] to find the best path and corresponding output codevectors. Different output codevectors in each subset can be considered as parallel transitions in the trellis or, as suggested in [12], one bit can be used to specify the trellis path and $R-1$ bits to specify symbols from the chosen subset. Using the aforementioned TCQ to quantize the source, a different trellis path might be chosen for quantizers working on different bit rates. So, even if the underlying $(R+1)$-bit quantizers are successively refinable, the structure of the trellis is not suitable for rate-scalability. In this section, we introduce a trellis structure which is scalable.

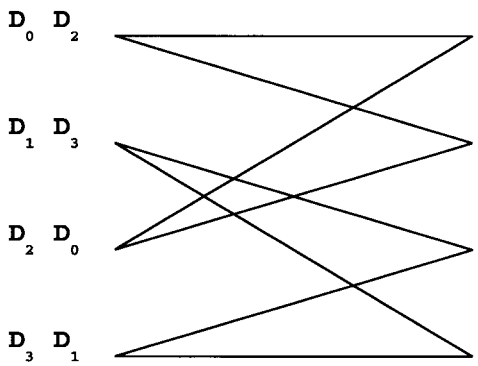

Fig. 1. Ungerboeck's four-state trellis.

To this end, we review the concept of the tensor product of trellises. Let T_1 and T_2 denote trellises with states $v_1, v_2, \cdots, v_{2^{b_1}}$, and $w_1, w_2, \cdots, w_{2^{b_2}}$, respectively. The tensor product trellis $T_1 \otimes T_2$ is a trellis with $2^{b_1+b_2}$ states $v_p \otimes w_q$, $p = 1, 2, \cdots, 2^{b_1}$, $q = 1, 2, \cdots, 2^{b_2}$. There is a transition between states $v_p \otimes w_q$ and $v_r \otimes w_s$ in $T_1 \otimes T_2$ if and only if there exist transitions between v_p and v_r in T_1 and between w_q and w_s in T_2. For J levels of refinement, we use $T_1 \otimes T_2 \otimes \cdots \otimes T_J$. The trellis $T_1 \otimes T_2$ is inherently scalable because each time section of $T_1 \otimes T_2$ is formed by replacing transitions of a time section of T_1 by copies of a time section of T_2.

Now, let us illustrate the construction of a scalable trellis by an example. We consider the four-state Ungerboeck's trellis T for both the first and second levels of refinement. The resulting trellis $T \otimes T$ is shown in Fig. 2 which is the tensor product of the trellis in Fig. 1 by itself. Each transition in Fig. 2 consists of two subtransitions. One subtransition is achieved by using the trellis of Fig. 1 for the first stage. Replacing the resulting path by another trellis, the second subtransition identifies a unique path in Fig. 2. Note that the survival path is derived in multistages and the decoding procedure of each stage can be done successively. As an example, let us enumerate the states in Figs. 1 and 2 from zero to 3 and from zero to 15, respectively. The transition from state 6 to state 9 in Fig. 2 consists of subtransitions from state 1 to state 2 and from state 2 to state 1 in Fig. 1 for the first and second stages, respectively. The corresponding partitions are D_{10} for the trellis in Fig. 2, D_1 for the trellis of the first stage, and D_0 for the trellis of the second stage. Note that the presentation of Fig. 2 is just to explain the underlying trellis and to shed some light on the structure of the overall trellis. In the process of encoding and decoding, there is no need to explicitly consider the trellis of Fig. 2. This is due to the fact that the encoding can

[1] While most of the ideas in this paper can be generalized to vectors (instead of scalars), in this work, we concentrate on scalars as reproduction values; however, with a slight abuse of terminology, we use the term "codevectors" instead of reproduction values.

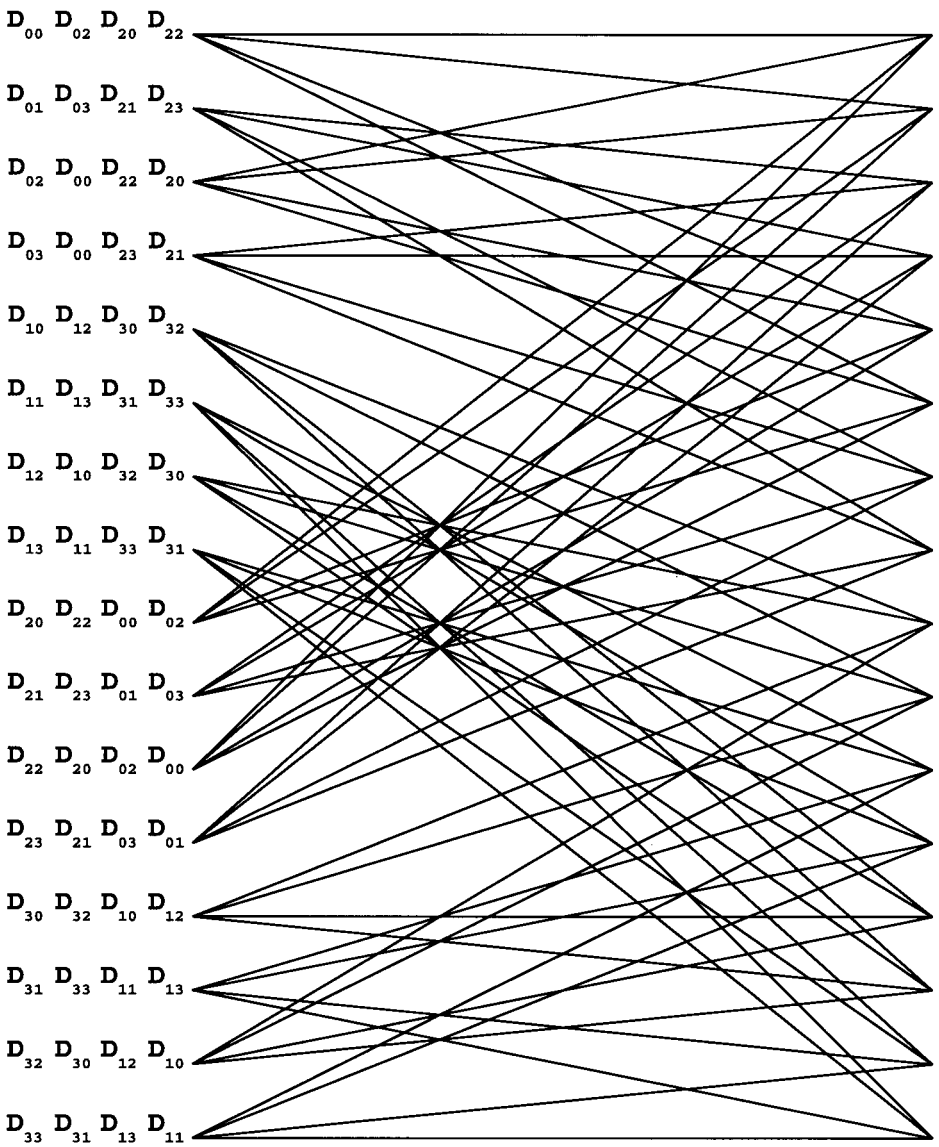

Fig. 2. The underlying trellis (two stages).

be done by following the definition of the tensor product trellis without explicitly using the trellis of Fig. 2. The structure of the trellis presented in this section is not restricted to two stages and more number of stages can be used in a similar fashion. Also, the same structure can be applied to trellises with higher number of states.

III. Successively Refinable Trellis Coded Quantization

In this section, we propose a TCQ structure which is successively refinable by using the trellis structure of Section II. Let us assume that each sample $x(n)$ is to be represented successively by the sequence of J reproductions $\tilde{\boldsymbol{x}}(n) = [\tilde{x}^1(n), \tilde{x}^2(n), \cdots, \tilde{x}^J(n)]$, $\tilde{\boldsymbol{x}}(n) \in \mathbb{R}^J$. Note that $\tilde{\boldsymbol{x}}(n)$ contains J representations of the same scalar sample, $x(n)$, and is not a J-dimensional vector of input samples. First r_1 bits are to be used to transmit $\tilde{x}^1(n)$. Then r_j bits are used to refine $\tilde{x}^{j-1}(n)$ and obtain $\tilde{x}^j(n)$, the jth description of $x(n)$ for $2 \le j \le J$. We refer to $\{r_j\}_{j=1}^J$ as the *incremental rates* and define the *aggregate rates*

$$\left\{ R_j = \sum_{m=1}^{j} r_m \right\}_{j=1}^{J}.$$

Let us define a set of codebooks $\{C_j\}_{j=1}^J$ where $C_j = \{c_i^j\}_{i=0}^{|C_j|-1}$, $|C_j| = 2^{R_j+j}$, and $|A|$ is the number of elements in set A. The structure of the first stage of the quantizer is similar to that of the TCQ proposed in [12] for rate r_1 with 2^{r_1+1} codevectors from C_1. Each codevector of the first stage is refined into 2^{r_2+1} codevectors and each transition of the trellis is replaced by a new trellis as was discussed in Section II. The same procedure is continued for other stages, $j = 2, \cdots, J$. A hierarchical set partitioning is used to preserve successive refinability. Let us assume that the codevectors in C_1 are in ascending order, i.e., $c_i^1 < c_{i'}^1 \Leftrightarrow i < i'$. Then

$$c_i^1 \in D_{i\%4}, \qquad i = 0, \cdots, |C_1| - 1$$

where $i\%4 = i - 4\lfloor i/4 \rfloor$ and $\lfloor a \rfloor$ is the largest integer smaller than or equal to a. Each codevector in C_{j-1}, $j = 2, \cdots, J$

corresponds to 2^{r_j+1} codevectors in C_j which are partitioned into four subsets as follows. If we enumerate the refinements of c_i^{j-1} in an ascending order as c_{ik}^j, $k=0,\cdots,2^{r_j+1}-1$, i.e., $c_{ik}^j < c_{ik'}^j \Leftrightarrow k<k'$, then

$$c_{ik}^j \in D_{l_1 l_2 \cdots l_j} \Leftrightarrow c_i^{j-1} \in D_{l_1 l_2 \cdots l_{j-1}} \quad \text{and} \quad k\%4 = l_j. \quad (1)$$

Figs. 2 and 3 show a specific example of our set partitioning and labeling. At any given trellis state of the jth refinement, the output codevector is selected from

$$D_{l_1 l_2 \cdots l_{j-1} 0} \cup D_{l_1 l_2 \cdots l_{j-1} 2}$$

or

$$D_{l_1 l_2 \cdots l_{j-1} 1} \cup D_{l_1 l_2 \cdots l_{j-1} 3}.$$

So, r_j bits are needed to uniquely define the output codevector at each state of the jth refinement. Defining an appropriate distortion measure as the cost function and using the Viterbi algorithm, the optimal path and codevector at each level of refinement are obtained. Choosing different cost functions results in different quantizers. For example, one may encode different stages separately or together. This will affect the complexity and performance of the quantizer. In following sections, we consider these issues more precisely.

IV. Design Algorithms

In this section, we define two cost functions and design the corresponding quantizers separately. First, we try to encode different stages sequentially. We adopt a greedy algorithm in which different stages are optimized one by one. We call this source coder tree-structured TCQ (TS-TCQ) since like a tree-structured vector quantizer (TSVQ) the decisions are made stage by stage through a tree structure.

Next, we consider an algorithm for the joint optimization of stages. The resulting quantizer is called successively refinable TCQ (SR-TCQ).

A. TS-TCQ

In TS-TCQ, different stages are encoded sequentially. A greedy algorithm is adopted to optimize the performance of different stages separately. The encoder of the first stage is that of an optimal TCQ at rate r_1. Let us define the per-sample MSE between input $X(n)$ and the jth refinement $\tilde{X}^j(n)$ as $\tilde{\mathcal{D}}_j = E[(X(n)-\tilde{X}^j(n))^2]$. The jth encoder, given the output of the first $(j-1)$ encoders, finds the jth output which minimizes $\tilde{\mathcal{D}}_j$. This can be considered as designing $|C_{j-1}|$ TCQ's at rate r_j which are optimal for the source samples which have been represented by $c_i^{j-1}, i=0,1,\cdots,|C_{j-1}|-1$.

The resulting quantizer is called TS-TCQ since the decisions are made stage by stage through a tree structure. Note that here unlike a TSVQ the jth refinement $\tilde{x}_j(n)$ is not necessarily the closest codevector of C_j to the input sample. In fact, the intervals corresponding to codevectors at the $(j-1)$st level of refinement overlap with each other. This is due to the fact that the same sample x might be quantized to different codevectors of a TCQ at different times (because of the trellis structure).

To design a TS-TCQ, we need to obtain a set of optimal codebooks, C_j, $j=1,\cdots,J$. Since the encoding procedure at each stage of a TS-TCQ is performed separately, codebooks can be designed sequentially. So, the following simple algorithm provides the optimal codebook (given the structure of TS-TCQ):

Design Algorithm for TS-TCQ's:

{1} Set $j=1$.

{2} Use a regular TCQ design algorithm to find the best codebook C_1 at rate r_1.

{3} Set $j=j+1$.

{4} Given the codebooks $C_m, \forall m < j$, find the best codebook C_j.

{5} If $j<J$, go to step {3}. Otherwise, stop.

Step {4} of the above algorithm needs further elaboration. However, since it is a special case of the design algorithm for SR-TCQ's, we do not explain it separately for the sake of brevity. The design algorithm for SR-TCQ's is presented next.

B. SR-TCQ

Now, we seek to minimize a weighted sum of the stage distortions $\tilde{\mathcal{D}}_j$'s. We introduce a random process $Q=\{q(n)\}_{n=1}^{\infty}$ with $q(n) \in \{1,2,\cdots,J\}$ and marginal probability mass function $p_Q(j)$ which selects one of the J reproductions $\tilde{x}^j(n)$, $j=1,2,\cdots,J$, to represent $x(n)$. The process Q can be used to explicitly model the probability that a given number of bits will be used to represent X, or may be used to weight the distortion at each level according to its relative importance in an application [10]. We do not consider the time-varying characteristics of $p_Q(j)$, if any, in our design algorithm. In other words, we design a quantizer for a given set of $p_Q(j)$ numbers. If $p_Q(j)$ changes in time, the user should adapt the parameters of the quantizer to the new values. We assume that Q and X are independent. Define $\hat{X}$ so that $q(n)=j \Leftrightarrow \hat{x}(n)=\tilde{x}^j(n)$. The goal is to minimize the MSE between X and $\hat{X}$, which is expressed as

$$\mathcal{D} = \sum_{j=1}^{J} p_Q(j)\tilde{\mathcal{D}}_j. \quad (2)$$

The minimization of the weighted average distortion $\mathcal{D}$ is subject to constraints on the bit rate available for transmission of X at each level. We call this approach successively refinable trellis-coded quantization. Unlike TS-TCQ which provides the output of different levels of refinement one by one, in SR-TCQ, *the output of all the stages are obtained together*. Note that although the trellis structure for TS-TCQ and SR-TCQ are similar, the encoding procedures are completely different. In TS-TCQ, encoder stages operate sequentially. The only information about the first $j-1$ stages of the TCQ which is needed to find the jth refinement of the input is the codeword of the $(j-1)$st level of refinement. On the other hand, SR-TCQ utilizes all J distortions computed in different J stages of the quantizer to find the best path in the trellis. In other words, TS-TCQ uses the idea of tree-structured quantizers which can be applied to any quantizer structure. In contrast, the definition of a scalable trellis and the way we calculate distortions in the Viterbi algorithm are essential in the construction of SR-TCQ.

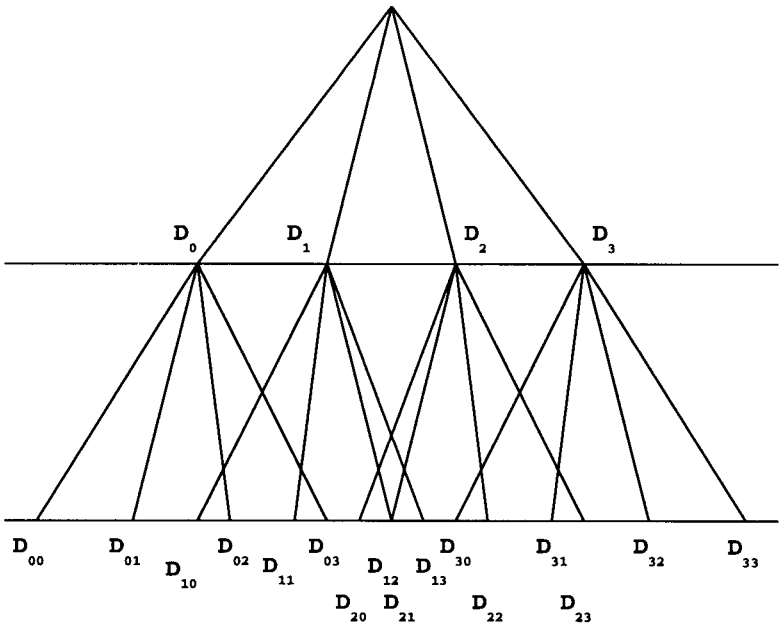

Fig. 3. A hierarchical set partitioning for a two-level SR-TCQ ($r_1 = r_2 = 1$ bit/sample).

For a training sequence with L samples, $x(l), l = 1, \cdots, L$, we can express the sample-average distortion as

$$\overline{\mathcal{D}} = \frac{1}{L} \sum_{j=1}^{J} p_Q(j) \sum_{l=1}^{L} [x(l) - \tilde{x}_j(l)]^2. \tag{3}$$

For a given set of codebooks, using the Viterbi algorithm with the distortion measure in (3) provides the best outputs at different levels of refinement. Let us define B_i^j as the set of all training samples which are encoded as c_i^j. Then if we replace each codevector c_i^j with a new codevector $\tilde{c}_i^j$ defined by

$$\tilde{c}_i^j = \frac{1}{|B_i^j|} \sum_{x(l) \in B_i^j} x(l) \tag{4}$$

the resulting set of codebooks provides a lower distortion when the same path and codewords which have been used for the old codevectors are utilized. Note that $x(l) \in B_i^j$ does not necessarily mean that c_i^j is the closest codevector to $x(l)$. Also, B_i^j's are not disjoint. The following algorithm can be used to design SR-TCQ's:

Design Algorithm for SR-TCQ's:

{0} Initialization:
- (a) Pick a small positive number ϵ.
- (b) Pick an initial set of codebooks..
- (c) Set $n = 1$ and $\overline{\mathcal{D}}^{(0)} = +\infty$.

{1} Encode the training sequence using the Viterbi algorithm and the distortion measure in (3). Denote the resulting distortion as $\overline{\mathcal{D}}^{(n)}$

{2} Update the codebooks by using (4) to find the best set of codevectors.

{3} If $\frac{\overline{\mathcal{D}}^{(n-1)} - \overline{\mathcal{D}}^{(n)}}{\overline{\mathcal{D}}^{(n)}} > \epsilon$, set $n = n + 1$ and go to {1}. Otherwise, stop.

Since the distortion is reduced at each step of the algorithm and the distortion is lower-bounded by zero, convergence is guaranteed.

C. Overlapping Intervals

In this subsection, we explain the reason for overlapping intervals in our hierarchical set partitioning. As an example, Fig. 3 demonstrates a set of codebooks for TS-TCQ and SR-TCQ ($r_1 = r_2 = 1$ bit/sample). As it is seen from Fig. 3, the resulting codevectors do not construct a tree-structured scalar quantizer with nonoverlapping intervals. For example, D_1 branches intervene D_0 and D_2 branches. This is due to the fact that the output codevector of the first stage is selected from $D_0 \cup D_2$ or $D_1 \cup D_3$. So, for example, a sample close to a codevector in D_1 may be encoded to a codevector in D_0 or *vice versa*. The crossing in Fig. 3 allows the encoder to fix such shortcomings at the second level of refinement. Another example is given in Fig. 4 to show the hierarchical set partitioning when the incremental rates are more than one. As can be seen in Fig. 4, the number of intervals corresponding to each path in the trellis is more than one which results in parallel transitions for trellis paths.

V. Computational Complexity

In this section, we compare the computational complexity of TCQ, MS-TCQ, TS-TCQ, and SR-TCQ for a four-state trellis with each other. The computational complexity of the decoders are more or less the same and much lower than the complexity of encoders. So, we only consider the encoders in our analysis. Let us assume that we only have two levels of refinement and each codebook is partitioned into four sets. So, for a doubled codebook TCQ with rate R, each partition contains 2^{R-1} codevectors corresponding to 2^{R-1} parallel branches. Since parallel branches come from the same node

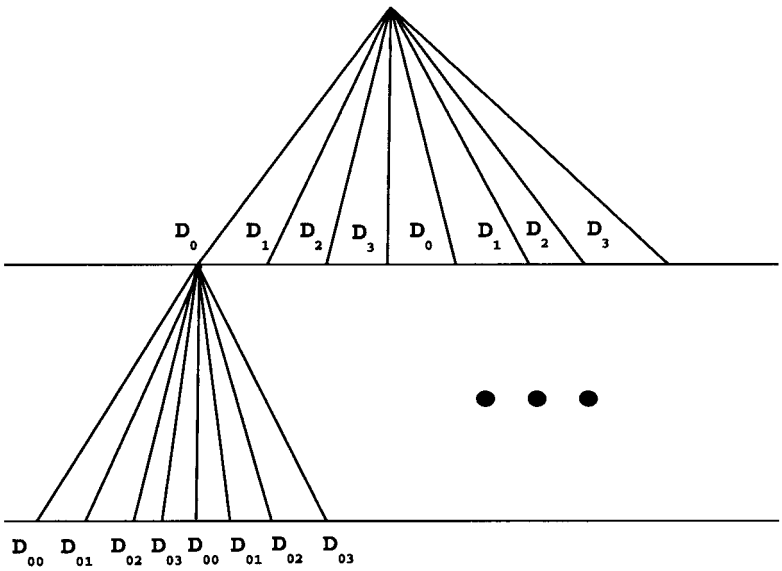

Fig. 4. A hierarchical set partitioning for a two-level SR-TCQ ($r_1 = r_2 = 2$ bits/sample).

TABLE I
SNR RESULTS IN DECIBELS FOR DIFFERENT QUANTIZERS (FOUR STATES); MEMORYLESS GAUSSIAN SOURCE

		TS-TCQ		SR-TCQ		MS-TCQ		TCQ		Lloyd-Max		D(R)	
r_1	r_2	D_1	D_2	D_1	D_2	D_1	D_2	D_1	D_2	D_1	D_2	$D(R_1)$	$D(R_2)$
1	1	5.00	10.11	4.97	10.37	N/A	N/A	5.00	10.55	4.40	9.30	6.02	12.04
1	2	5.00	15.66	4.99	15.82	N/A	N/A	5.00	16.21	4.40	14.62	6.02	18.06
2	1	10.55	15.96	10.52	16.21	N/A	N/A	10.55	16.21	9.30	14.62	12.04	18.06
1	3	5.00	21.30	5.00	21.42	4.29	20.05	5.00	21.91	4.40	20.22	6.02	24.08
2	2	10.55	21.55	10.54	21.69	10.24	19.70	10.55	21.91	9.30	20.22	12.04	24.08
3	1	16.21	21.76	16.18	21.99	15.89	19.73	16.21	21.91	14.62	20.22	18.06	24.08

TABLE II
SNR RESULTS IN DECIBELS FOR DIFFERENT QUANTIZERS (FOUR STATES); MEMORYLESS LAPLACIAN SOURCE

		TS-TCQ		SR-TCQ		TCQ		Lloyd-Max		D(R)	
r_1	r_2	D_1	D_2	D_1	D_2	D_1	D_2	D_1	D_2	$D(R_1)$	$D(R_2)$
1	1	4.39	9.59	4.37	9.79	4.39	9.45	3.01	7.54	6.62	12.66
1	2	4.39	14.98	4.39	15.09	4.39	14.87	3.01	12.64	6.62	18.68
2	1	9.45	15.20	9.42	15.45	9.45	14.87	6.62	12.64	12.66	18.68
1	3	4.39	20.47	4.39	20.54	4.39	20.53	3.01	18.12	6.62	24.71
2	2	9.45	20.70	9.45	20.83	9.45	20.53	6.62	18.12	12.66	24.71
3	1	14.87	20.86	14.86	21.06	14.87	20.53	12.66	18.12	18.68	24.71

(same survival distortion), we can pick the best candidate among them by comparing the value of the input sample to $2^{R-1}-1$ thresholds (like the encoder of an $(R-1)$-bit scalar quantizer which requires $R-1$ comparisons). After picking the best candidate for each partition, the corresponding distortion is calculated which is used in the Viterbi algorithm. So, we need one multiplication and one addition per parallel branches per state which is *independent* of the rate. For a four-state TCQ operating at rate R, we need four multiplications, 12 additions, and $4R$ comparisons per input sample. The same number of operations is needed for each stage of TS-TCQ or MS-TCQ (R should be replaced by r_j for each of

TABLE III
SNR RESULTS OF SR-TCQ2 IN DECIBELS; MEMORYLESS GAUSSIAN SOURCE

		SR-TCQ2		TCQ(R_2)	
r_1	r_2	D_1	D_2	4 states	16 states
1	1	4.28	10.81	10.56	10.78
1	2	4.24	16.55	16.19	16.40
2	1	9.39	16.77	16.19	16.40
1	3	3.98	22.49	21.91	22.13
2	2	9.23	22.56	21.91	22.13
3	1	15.17	22.57	21.91	22.13

TABLE IV
SNR RESULTS OF SR-TCQ2 IN DECIBELS; MEMORYLESS LAPLACIAN SOURCE

		SR-TCQ2		TCQ(R_2)		
r_1	r_2	D_1	D_2	4 states	16 states	Quadrupled (16 states)
1	1	3.36	10.33	9.45	9.69	10.47
1	2	3.55	15.83	14.87	15.16	16.20
2	1	8.66	15.99	14.87	15.16	16.20
1	3	3.03	21.64	20.53	20.71	21.76
2	2	8.33	21.75	20.53	20.71	21.76
3	1	12.49	21.87	20.53	20.71	21.76

the j stages). For an SR-TCQ, the encoding decisions are made simultaneously for both stages. So, the underlying trellis contains 16 states. We need 16 multiplications, 80 additions, and $4r_1+16r_2+28$ comparisons per input samples. One of the advantages of TCQ over other quantization schemes is the fact that its computational complexity is roughly independent of the rate [12]. Most other quantization schemes (excluding scalar quantizers) suffer from an exponential growth in computational burden with the rate. The rate-scalable TCQ's proposed in this work preserve the complexity advantage of a TCQ, i.e., their computational complexity is *roughly independent* of the encoding rates.

VI. SIMULATION RESULTS AND CONCLUSIONS

In this section, we present simulation results and compare our results with those of TCQ, MS-TCQ, Lloyd–Max quantizer, and the distortion-rate function for Gaussian and Laplacian memoryless sources. We present results for TS-TCQ's and SR-TCQ's with two levels of refinement (SR-TCQ is designed for two equiprobable stages). The proposed algorithms are not restricted to two stages; however, having two stages simplifies the presentation and allows a clear discussion on results. Tables I and II show the R-D performance of different quantizers (four-state trellis) for zero-mean, unit-variance memoryless Gaussian and Laplacian sources, respectively. The acronym N/A stands for not available throughout the tables. The TS-TCQ results are better than those of MS-TCQ. The signal-to-noise ratio (SNR) difference is more than 2 dB in one case. The computational complexity is the same although the memory requirements of TS-TCQ is more than that of MS-TCQ. The performance of the second stage of SR-TCQ is a few tenths of a decibel better than that of TS-TCQ while the first stage of SR-TCQ performs almost as well as the first stage of TS-TCQ. One interesting observation is the fact that for a fixed $R_2 = r_1 + r_2$, by increasing the bit rate of the first stage r_1, the performance of the second stage of TS-TCQ and SR-TCQ is improved. This is not a trend for MS-TCQ. Also, note that there is always a combination of rates for which SR-TCQ outperforms TCQ at rate R_2 (although the comparison is not completely fair since SR-TCQ is more complex than TCQ).

We also provide the simulation results for a two-stage SR-TCQ which uses $\tilde{\mathcal{D}}_2$ instead of $\mathcal{D}$ as the distortion measure in Tables III and IV. The performance of the resulting quantizer, denoted SR-TCQ2, is identical to that of a TCQ using the trellis of Fig. 2 although the encoding and decoding processes are different and SR-TCQ2 provides an embedded bit stream. The motivation behind SR-TCQ2 is the fact that some of the results in Tables I and II are so good that makes the study of the best possible last stage performance of an SR-TCQ and its comparison with a nonscalable TCQ interesting. To have a more conclusive comparison, we also provide the performance of TCQ's with four and 16 states. Tables III and IV show that not only does SR-TCQ2 provide some degrees of rate-scalability, but also its performance is better than that of a 16-state doubled codebook TCQ at all reported rates. Note that the computational complexity of SR-TCQ2 is less than that of SR-TCQ because there is no need to calculate the distortions of the intermediate stages and multiply them by different weights. For the Laplacian source, the results of SR-TCQ2 are much better than those of a doubled codebook TCQ. This is due to the fact that doubling the codebook is not enough for a Laplacian source [12]. Quadrupled codebooks provide a better performance while increasing the computational complexity almost by a factor of two [12]. In Table IV, we also tabulate the performance of the best quadrupled codebook results from [12]. A similar quadrupled codebook can be used in the design of SR-TCQ's.

The proposed algorithms may be extended to trellis-coded vector quantizers presented in [18]. Another future work includes the design of entropy constrained successively refinable trellis-coded quantizers. Such an extension is achievable by combining the schemes proposed in [10] and the trellis intro-

TABLE V
OPTIMIZED CODEVECTORS OF SR-TCQ2
FOR THE MEMORYLESS GAUSSIAN SOURCE

$r_1 = 1, r_2 = 1$			
D_0	D_1	D_2	D_3
-1.10	-0.44	0.44	1.10
-2.18 -1.46	-1.25 -0.65	-0.18 0.23	0.52 0.96
-0.96 -0.52	-0.22 0.19	0.65 1.25	1.46 2.18
$r_1 = 1, r_2 = 2$			
D_0	D_1	D_2	D_3
-1.11	-0.44	0.44	1.11
-2.73 -2.11	-1.58 -1.09	-0.41 -0.07	0.36 0.68
-1.71 -1.32	-0.80 -0.52	0.06 0.33	0.84 1.13
-1.13 -0.83	-0.33 -0.06	0.53 0.81	1.32 1.71
-0.68 -0.34	0.07 0.40	1.09 1.58	2.12 2.73

duced in Section II. Entropy constrained successively refinable trellis-coded quantizers are applicable to image coding systems proposed in [13], [15], and [19]. Based on the superb results reported in [13], [15], and [19], we believe that such a combination provides a rate-scalable image coding system with very good performances.

Using the trellis of Section II to design scalable trellis-coded modulation schemes is another direction to extend this work. Preliminary results show that such a system is suitable for broadcast applications.

To have a better visualization of Figs. 3 and 4 and the corresponding arguments in Section IV, we tabulate the optimized codevectors of SR-TCQ2 for the memoryless Gaussian source in Table V.

REFERENCES

[1] R. V. Cox, J. Hagenauer, N. Seshadri, and C.-E. Sundberg, "Subband speech coding and matched convolutional channel coding for mobile radio channels," *IEEE Trans. Signal Processing*, vol. 39, pp. 1717–1731, Aug. 1991.

[2] H. Jafarkhani, P. Ligdas, and N. Farvardin, "Adaptive rate allocation in a joint source/channel coding framework for wireless channels," in *Proc. Vehicular Technology Conf.* (Atlanta, GA, Apr. 1996), pp. 492–496.

[3] V. Chande, H. Jafarkhani, and N. Farvardin, "Joint source-channel coding of images for channels with feedback," in *IEEE Information Theory Workshop* (San Diego, CA, Feb. 1998).

[4] V. Koshelev, "Hierarchical coding of discrete sources," *Probl. Pered. Inform.*, vol. 16, no. 3, pp. 31–49, 1980.

[5] ———, "An evaluation of the average distortion for discrete scheme of sequential approximation," *Probl. Pered. Inform.*, vol. 17, no. 3, pp. 20–33, 1981.

[6] R. M. Gray, "Source coding a binary symmetric source over a simple network," in *Proc. Hawaii Int. Conf. System Sciences*, Jan. 1973, pp. 354–355.

[7] W. H. R. Equitz and T. M. Cover, "Successive refinement of information," *IEEE Trans. Inform. Theory*, vol. 37, pp. 269–274, Mar. 1991.

[8] B. Rimoldi, "Successive refinement of information: Characterization of the achievable rates," *IEEE Trans. Inform. Theory*, vol. 40, pp. 253–259, Jan. 1994.

[9] R. V. Cox, S. L. Gay, Y. Shoham, S. Quackenbusch, N. Seshadri, and N. S. Jayant, "New directions in subband coding," *IEEE J. Select. Areas Commun.*, vol. 6, pp. 391–409, Feb. 1988.

[10] H. Brunk, H. Jafarkhani, and N. Farvardin, "Design of successively refinable scalar quantizers," preprint.

[11] C. F. Barnes and R. L. Frost, "Residual vector quantizers with jointly optimized code books," *IEEE Trans. Inform. Theory*, vol. 39, pp. 565–580, Mar. 1993.

[12] M. W. Marcellin and T. R. Fischer, "Trellis coded quantization of memoryless and Gauss–Markov sources," *IEEE Trans. Commun.*, vol. 38, pp. 82–93, Jan. 1990.

[13] R. L. Joshi, H. Jafarkhani, J. H. Kasner, T. R. Fischer, N. Farvardin, M. W. Marcellin, and R. H. Bamberger, "Comparison of different methods of classification in subband coding of images," *IEEE Trans. Image Processing*, vol. 6, pp. 1473–1486, Nov. 1997.

[14] H. A. Aksu and M. Salehi, "Multi-stage trellis coded quantization (MS-TCQ)," in *Proc. Conf. Information Sciences and Systems* (Baltimore, MD, Mar. 1995).

[15] P. J. Sementilli, A. Bilgin, F. Sheng, M. W. Marcellin, and J. C. Kieffer, "Progressive transmission in trellis coded quantization-based image coders," in *Proc. Int. Conf. Image Processing* (Santa Barbara, CA, Oct. 1997).

[16] G. Ungerboeck, "Channel coding with multilevel/phase signals," *IEEE Trans. Inform. Theory*, vol. IT-28, pp. 55–67, Jan. 1982.

[17] G. D. Forney, Jr., "The Viterbi algorithm" (Invited Paper), *Proc. IEEE*, vol. 61, pp. 268–278, Mar. 1973.

[18] T. R. Fischer, M. W. Marcellin, and M. Wang, "Trellis coded vector quantization," *IEEE Trans. Inform. Theory*, vol. 37, pp. 1551–1566, Nov. 1991.

[19] H. Jafarkhani and N. Farvardin, "Adaptive image coding using spectral classification," *IEEE Trans. Image Processing*, vol. 7, pp. 605–610, Apr. 1998.