Correspondence

Predictive Multistage Vector Quantizer Design Using Asymptotic Closed-Loop Optimization

Hosam Khalil and Kenneth Rose

***Abstract*—This correspondence builds on the asymptotic closed-loop approach to predictive vector quantizer design [1], and extends it to the design of predictive multistage vector quantizers for low bit rate video coding. The design approach resolves longstanding shortcomings, in particular, design stability and empty-cell problems. Simulation results show substantial gains over traditional design approaches.**

***Index Terms*—Multistage vector quantization, predictive quantizer design, splitting.**

I. Introduction

DIGITAL video data transmission over communication channels with limited bandwidth requires the use of lossy compression techniques. Low bit rates, particularly in the range of 16-32 kbits/s, are gaining in importance due to the Internet and wireless mobile communications. Compression to achieve bit rates at this range is considerably difficult, and it is not known with certainty which fundamental compression technique is best suited for this task. The most widely used approach to efficient video compression is based on motion compensation, prediction and quantization. The prediction residual is usually handled as a two-dimensional signal, much as if it were a still image. The predominant residual compression approach involves application of the discrete cosine transform (DCT), which is the case in the major standards such as H.263 and MPEG.

Another approach to coding the prediction residual involves vector quantization (VQ). Generally, there are several arguments in support of VQ for video compression. Shannon's theory implies that vector quantizers are asymptotically optimal, where asymptotic here is in terms of vector length (typical blocks in video coding correspond to long vectors.) Another important argument is that VQ is a very general framework and includes DCT compression as a special constrained case [2]. Thus, it may be argued that DCT can not outperform the best VQ.

However, there are two main objections to the use of VQ in video coding. The first objection is concerned with complexity: VQ complexity grows exponentially with the product of vector dimension and rate. This problem can be alleviated by one of several possible techniques to reduce search or memory complexity. Examples include classified VQ (CVQ), tree-structured VQ (TSVQ), and multistage VQ (MSVQ) [2]. Multistage VQ has been found to be a particularly efficient technique, with acceptable search and memory complexity. It has been widely adopted in speech coding standards and, more recently, there have been reports on its applicability to image coding [3], [4]. We believe that adoption of MSVQ in video coding (see Fig. 1) may be beneficial, especially at low bit rates. It should further be mentioned that even though the encoding complexity of a VQ-based system is typically higher than that of H.263, the decoding complexity is significantly lower, as it consists of simple table look-up operations. This feature is particularly important in streaming applications.

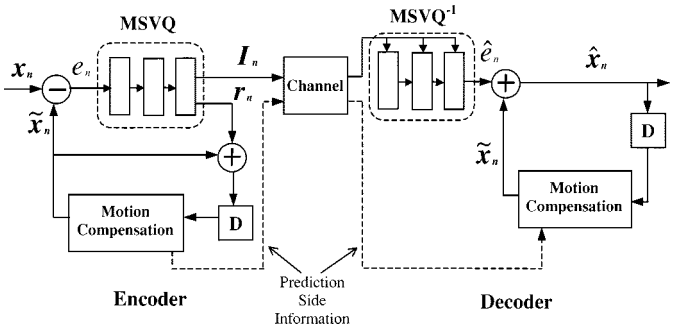

Fig. 1. Basic predictive video coding system based on a multistage quantizer.

The second objection appears in prediction-based applications: Predictive quantizer design is problematic, due to the destabilizing effect of the prediction loop, and the design often fails to produce an optimal, or even a good, quantizer. We previously proposed a solution to this problem in [1] and [5], for the simpler case of single-stage predictive vector quantizers (PVQ), which offers several attractive features, and achieves significant gains over traditional design methods. Moreover, the coding performance was competitive with H.263.

An additional pitfall of existing PVQ design algorithms is the "empty-cell" problem, where an encoding cell loses its training data to other cells. This phenomenon is aggravated by any instability in the iterative design. In this correspondence, we also propose a selective splitting codebook design algorithm that grows codebooks gradually and optimally. The proposed selective splitting design is applied to variable rate multistage vector quantizers, and is found to be effective when incorporated into the predictive quantizer design approach [1]. Simulation results for video coding are provided.

II. Predictive Quantizer Design

Obtaining a suitable training set for predictive quantizer design, which accurately represents the true signal statistics, is quite a challenging objective. In contrast with the standard quantizer, the predictive quantizer operates on the prediction error. But since the prediction is based on the reconstruction of past vectors (previous reconstructed frames in the case of video), the prediction error depends on the quantizer itself. The effective training set, i.e., the sequence of prediction errors, is thus not fixed but changes whenever the quantizer is modified, causing a complex interaction between the quantizer and the training set used for its design. In this section, we give a short overview of various predictive quantizer design approaches.

A. Open-Loop Approach

In [6], two techniques were introduced for predictive quantizer design and have been widely used since. The first technique, called the open-loop (OL) approach, is the simplest. A training set of prediction errors is extracted directly from the original, unquantized source vectors. The approach is called "open-loop" because the reconstructed vec-

Manuscript received May 18, 2000; revised August 20, 2001. This work was supported in part by the National Science Foundation under Grants MIP-9707764 and EIA-9986057, the University of California MICRO Program, Conexant Systems, Inc., Dolby Laboratories, Inc., Lucent Technologies, Inc., MedioStream, Inc., and Qualcomm, Inc. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. A. Enis Cetin.

The authors are with the Department of Electrical and Computer Engineering, University of California, Santa Barbara, CA 93106 USA (e-mail: rose@ece.ucsb.edu).

Publisher Item Identifier S 1057-7149(01)09572-0.

tors are not fed back through the predictor. Given a set of source vectors, X: $\{x_n\}_{n=0}^{N}$, a training set of prediction error vectors is generated, $\boldsymbol{T} = \{e_n\}_{n=1}^{N}$, where $e_n = x_n - \tilde{x}_n = x_n - P[x_{n-1}]$, and where P denotes prediction. The entire training set can be obtained in parallel, as there are no closed-loop dependencies. The quantizer $\boldsymbol{Q}_{OL}$ is designed using the fixed training set $\boldsymbol{T}$ and an iterative algorithm such as the generalized Lloyd algorithm (GLA [7]).

B. Closed-Loop Approach

The closed-loop approach (CL) appeared in several variants, including its original version in [6] and the so-called "semi-closed-loop" method in [2]. Both approaches use a real closed-loop system to generate the prediction errors in an iterative fashion. Given a quantizer at iteration $i-1$, which we denote by $\boldsymbol{Q}^{(i-1)}$, a training set of prediction errors is generated for iteration i, $\boldsymbol{T}^{(i)} = \{e_n^{(i)}\}_{n=1}^{N}$ where, $e_n^{(i)} = x_n - P[\hat{x}_{n-1}^{(i)}]$, and $\hat{x}_n^{(i)} = P[\hat{x}_{n-1}^{(i)}] + \boldsymbol{Q}^{(i-1)}(e_n^{(i)})$. Note that we must alternate between the computation of $e_n^{(i)}$ and $\hat{x}_n^{(i)}$ for $n = 1, 2, \ldots, N$. Specifically, given the resulting set of prediction errors, $\boldsymbol{T}^{(i)}$, a new quantizer, $\boldsymbol{Q}^{(i)}$, is designed. Next, a new sequence of prediction errors is generated for iteration $i+1$, and so on. The initial quantizer, $\boldsymbol{Q}^{(0)}$, is usually chosen to be the outcome of the OL method.

Another approach to predictive quantizer design in [8] was based on gradient approaches to jointly optimize quantizer and predictor. However, in video coding applications, the predictor takes on the form of motion compensation and thus need not be optimized in the sense of [8].

C. Asymptotic Closed-Loop Approach

The main advantage of OL is that the training set, $\boldsymbol{T}$, is fixed, and we can design the quantizer by applying a standard optimization technique. Since the training set remains unchanged, the design algorithm is ensured to converge to a locally optimal solution. However, during the actual operation of the compression system, prediction must be performed using reconstructed source vectors. This causes a statistical mismatch, which is compounded by error build-up via feedback through the prediction loop, and results in poor performance.

On the other hand, the CL training residuals are generated by the same closed-loop coder that will be used in actual operation of the system. Thus, the input residual error statistics are expected to be similar to those used to train the quantizer. However, convergence of the algorithm is not guaranteed, as the training set changes per iteration in an unpredictable fashion, and hence the training algorithm must be closely monitored to detect instability.

Motivated by such observations, we proposed the Asymptotic Closed-Loop (ACL) [1] approach, which offered significantly improved design stability leading to improved overall coding performance. ACL capitalizes on the merits of OL and CL while circumventing their drawbacks; it inherits the design stability of open-loop techniques while ultimately optimizing the system for closed-loop operation. The ACL procedure for quantizer design is explained next.

The main objective of the design procedure is to avoid accumulation of errors due to mismatched quantization through the prediction loop. Therefore, we base our prediction on the reconstructed vectors of the *previous iteration.* The training set is, in effect, generated by

$$e_n^{(i)} = x_n - P\left[\hat{x}_{n-1}^{(i-1)}\right], \qquad n = 1, 2, \ldots, N. \tag{1}$$

Having collected the set of training vectors we optimize a new quantizer, $\boldsymbol{Q}^{(i)}$. The resulting quantizer is then used to generate the new set of reconstruction vectors

$$\hat{x}_n^{(i)} = P\left[\hat{x}_{n-1}^{(i-1)}\right] + \boldsymbol{Q}^{(i)}\left(e_n^{(i)}\right), \qquad n = 1, 2, \ldots, N. \tag{2}$$

A major difference between ACL and CL is in the way the residuals and reconstructions are generated in each iteration. In CL, we must alternate between the computation of $e_n^{(i)}$ and $\hat{x}_n^{(i)}$ for each of $n = 1, 2, \ldots, N$, before we proceed to the design of the quantizer. This is dictated by the closed-loop operation of the feedback system. In contrast, the ACL approach requires us to first calculate $e_n^{(i)}$ for *all* $n = 1, 2, \ldots, N$, then design the quantizer, and finally calculate $\hat{x}_n^{(i)}$ for *all* $n = 1, 2, \ldots, N$.

Note that the quantizer $\boldsymbol{Q}^{(i)}$ is used to encode *exactly* the same prediction error vectors used for its design. Neglecting possible problems of local optima, this is the best quantizer for these vectors. We are thus assured that the resulting reconstruction is improved, and this results in better prediction for the next iteration. Subject to the high-rate assumption that smaller prediction errors lead to smaller quantization errors, we obtain monotonic improvement throughout the process.

It is emphasized that the entire design is in open-loop mode since we compute prediction errors for all elements of the sequence before quantization. The consequence of using a fixed reconstructed set for the prediction in each iteration becomes apparent by studying the different phases of the training procedure. At each iteration, the reconstructed set, on which the prediction for the next iteration will be based, is generated by applying the optimized quantizer and predictor based on the previous fixed set. Since the new reconstructed set will better approximate the original input sequence, the distortion at each iteration is generally decreasing, and we expect the process to converge. At convergence, further iterations do not modify the training set. The quantizer is hence assumed to have converged, i.e., $\boldsymbol{Q}^{(i+1)} = \boldsymbol{Q}^{(i)}$, which immediately ensures that the reconstruction sequence is unchanged, i.e., $\hat{x}_n^{(i+1)} = \hat{x}_n^{(i)}$, as well as the next-frame prediction sequence $P[\hat{x}_{n-1}^{(i)}] = P[\hat{x}_{n-1}^{(i-1)}]$. This implies that the prediction may equivalently be based on the reconstruction of the current iteration, instead of on the reconstruction from the previous iteration. The procedure is thus open-loop in nature, yet it asymptotically converges to optimization of the closed-loop performance. The ACL procedure for the case of PMSVQ design for video is illustrated in Fig. 2.

III. Selective Splitting Quantizer Design

The design instability problems of the CL method are more pronounced in the case of MSVQ. Training vectors that have accumulated considerable error (as a result of the closed loop) will enforce undesirable modification of the codebooks in an effort to accommodate outliers. More specifically, the objective of the training algorithm is to adjust the parameters of the quantizer so as to minimize the training distortion. When vectors with excessive error accumulation are included with the training set, the design algorithm will attempt to minimize the coding distortion by allocating precious resources to unrepresentative vectors. In MSVQ, the interaction between the stages results in further sensitivity of stage codebooks to such misguided training set updates. The instability of the design is clearly apparent as the degraded codebooks will produce greater errors in the next iteration which, in turn, will further corrupt the codebooks. Moreover, this instability is a major contributing factor to the empty cell phenomenon that undermines the design procedure. The approach presented next is largely motivated by the realization that a stable and effective training procedure for predictive MSVQ (PMSVQ) is imperative for an efficient and competitive coder.

The structure of the MSVQ encoder [9] consists of a cascade of VQ stages as shown in Fig. 3. For a P-stage MSVQ, a pth stage quantizer Q_p, $p = 1, 2, \ldots, P$, is associated with a stage codebook C_p. Each stage codebook C_p contains N_p stage codevectors $\boldsymbol{y}_{p,n}$, where $n = 1, 2, \ldots, N_p$. The set of stage quantizers $\{Q_1, Q_2, \ldots, Q_P\}$ are equivalent to a single quantizer $\boldsymbol{Q}$, which is referred to as the di-

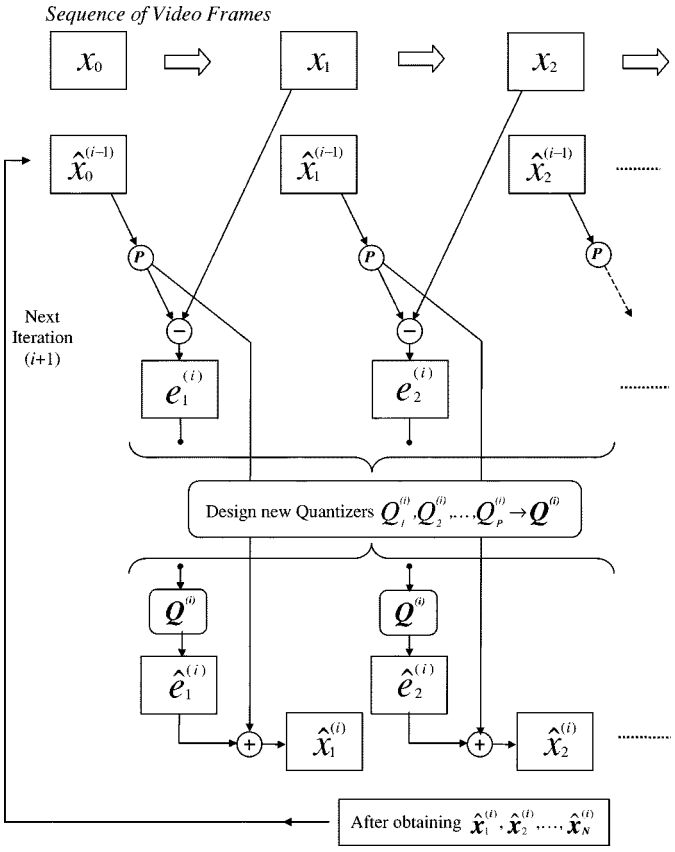

Fig. 2. Proposed ACL procedure: x_n denotes the original frame n, $\hat{x}_n^{(i)}$ denotes the n^{th} reconstructed frame at iteration i, and $e_n^{(i)}$ denotes the nth prediction error at iteration i. $\boldsymbol{Q}^{(i)}$ is the MSVQ trained on the prediction error sequence from iteration i, and $\hat{e}_n^{(i)}$ is the reconstructed prediction error after quantization by $\boldsymbol{Q}^{(i-1)}$. Note that the newly designed $\boldsymbol{Q}^{(i)}$ is used in the same iteration i to generate new reconstructed vectors in preparation for the next iteration $i+1$. The main difference between this design and the CL design is that there is *no feedback*; quantized prediction error is not fed back into the closed-loop system.

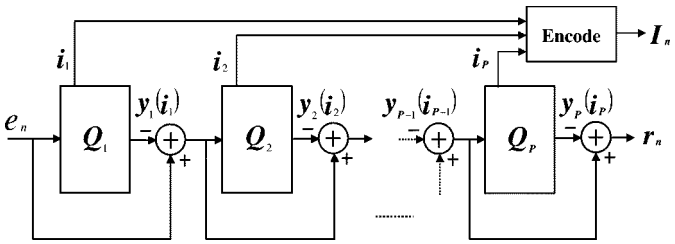

Fig. 3. Block diagram of a multistage vector quantizer encoder.

rect-sum vector quantizer. The best set of stage codevectors to represent an input vector can only be obtained with an exhaustive search. A well-known and efficient approach for suboptimal search is the M-search [10], and has been found to provide nearly optimal performance at a small fraction of the complexity.

The target application of this work is low bit rate video coding, and we must account for variation in local signal statistics. It is well known that variable rate coders can adapt to changing statistics, and offer higher compression efficiency than fixed rate coders. Hence, we propose to design a variable rate PMSVQ system for video compression. In this section, we start with a brief review of traditional approaches to fixed rate as well as entropy constrained multistage vector quantizer (EC-MSVQ) design. We then propose a more efficient design approach based on selective splitting. Finally, we embed this technique within the ACL method of Section II to obtain the proposed EC-PMSVQ design method.

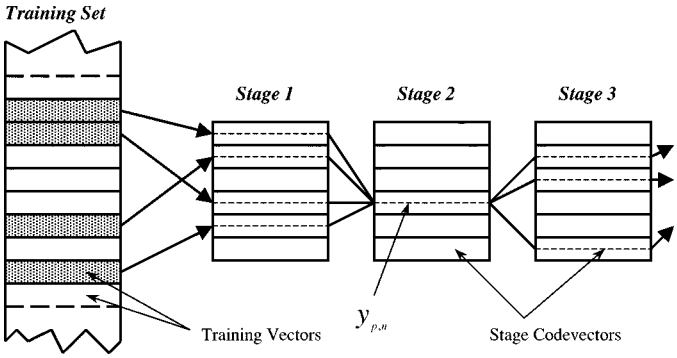

Fig. 4. Training subset for use in re-optimization of $\boldsymbol{y}_{p,n}$. Only the shaded vectors in training set use stage codevector $\boldsymbol{y}_{p,n}$ ($p = 2$, $n = 4$, in this example).

A. Overview of MSVQ and EC-MSVQ Design

Let us focus, for now, on the design of a nonpredictive MSVQ. The procedure is initialized with a set of P stage codebooks that are typically obtained by traditional sequential design [9]. It then iterates over the stages and optimizes each stage codebook while keeping the remaining stages fixed [11]. The re-optimization of quantizer Q_p consists of updating its codebook. The update of codevector $\boldsymbol{y}_{p,n}$ involves only the training subset $\boldsymbol{\chi}_{\boldsymbol{p,n}} \subset \boldsymbol{X}$ that selects $\boldsymbol{y}_{p,n}$ for its stage p codevector (see Fig. 4). It is convenient to remove the effect of all fixed stages from the training subset. Hence, for each training vector $\boldsymbol{x} \in \boldsymbol{\chi}_{\boldsymbol{p,n}}$, we subtract the fixed codevectors' contribution

$$\varphi(\boldsymbol{x}) = \boldsymbol{x} - (\boldsymbol{Q}(\boldsymbol{x}) - \boldsymbol{y}_{p,n}) \tag{3}$$

where $\boldsymbol{Q}(\boldsymbol{x})$ denotes an M-search quantization of $\boldsymbol{x}$ using the current MSVQ. The outcome of this operation is a new training set $\phi_{p,n} = \{\varphi(\boldsymbol{x}),\ \boldsymbol{x} \in \boldsymbol{\chi}_{\boldsymbol{p,n}}\}$ where the fixed codevectors' influence has been eliminated.

It is easy to see that minimum distortion is achieved by adjusting $\boldsymbol{y}_{p,n}$ to the centroid of $\phi_{p,n}$

$$\boldsymbol{y}_{p,n_{new}} = \frac{1}{|\phi_{p,n}|} \sum_{\varphi \in \phi_{p,n}} \varphi. \tag{4}$$

Noting further that $|\phi_{p,n}| = |\boldsymbol{\chi}_{\boldsymbol{p,n}}|$, we may equivalently write the update rule

$$\Delta \boldsymbol{y}_{p,n} = \frac{1}{|\boldsymbol{\chi}_{\boldsymbol{p,n}}|} \sum_{\boldsymbol{x} \in \boldsymbol{\chi}_{\boldsymbol{p,n}}} (\boldsymbol{x} - \boldsymbol{Q}(\boldsymbol{x})). \tag{5}$$

This rule has a satisfying simple interpretation. The stage codevector $\boldsymbol{y}_{p,n}$ is chosen such that the expected reconstruction error of the corresponding training subset is zero-mean (unbiased). After optimizing all codevectors in stage p, other stages are similarly optimized in order, and then the whole procedure is repeated until the rate of change falls below a prescribed threshold.

Design for entropy constrained MSVQ (EC-MSVQ) is an even more challenging problem. Joint optimization of the stages, where each stage design is performed using the ECVQ design method, was proposed in [12]. As with standard ECVQ design [13], a Lagrangian formulation is employed, where the encoding cost is a function of distortion and encoding rate: $L = D + \lambda R$. The Lagrangian multiplier, λ, controls the rate-distortion tradeoff, and R is estimated by the reconstruction entropy. The joint entropy of the VQ stages may be decomposed

into smaller components by introducing conditional probabilities between stages. First-order conditioning is typically assumed for simplicity, though optimality requires, in general, conditioning on all prior stages.

The standard EC-MSVQ design starts with a fixed rate ($\lambda = 0$) MSVQ and modifies it into a variable rate MSVQ by increasing λ in a series of steps. There are two drawbacks to standard design of EC-MSVQ: The computational complexity is considerable if the codebooks are large, and the final EC-MSVQ heavily depends on the initialization because the optimization may easily get trapped in a poor local minimum.

To reduce the complexity of EC-MSVQ, the pairwise nearest neighbor (PNN) algorithm [14] was extended to entropy constrained quantizer optimization by Finamore *et al.* [15] and then to EC-MSVQ by Kossentini *et al.* in [3]. In [15], the design procedure for single-stage quantizers uses the entire training set as the initial codebook, and recursively merges the pair of reproduction vectors that yields the least increase in distortion, until the desired codebook size is reached. In [3], the algorithm is extended to the case of multistage vector quantizers, where a fixed rate MSVQ is used as initialization then a modified PNN algorithm is consequently applied. The reduction in complexity, which only affects the design stage, is normally achieved at the cost of some degradation in subjective and objective performance of the quantizer.

B. Selective Splitting Approach to EC-MSVQ Design

To overcome complexity and initialization problems, we instead propose a selective splitting approach to the design of entropy constrained quantizers. Our objective is two-fold: 1) *directly* optimize the codebook to operate at the desired rate/distortion tradeoff, in contrast to [13], and thereby reduce complexity and 2) improve the initialization. An early use of the notion of splitting in the design of a fixed rate VQ was already made by Linde *et al.* [7]. Splitting may be viewed as the logical reverse of PNN, as selected codevectors are recursively split. The splitting mechanism we consider is closely related to the greedy splitting approach of Riskin and Gray [16], which was developed for the design of tree-structured VQ. However, here we use splitting as a means to improve the initialization and reduce complexity, and not for imposing a structure on the solution. We have used the selective splitting approach for designing standard variable rate quantizers [17] and found it to outperform standard GLA-based designs of [7], [12], and [13].

The proposed procedure for EC-MSVQ design is as follows.

Step 1) Initialize the P stage codebooks, each with a single vector. Set $N_p = 1,\ p = 1,\ \ldots,\ P$.

Step 2) For all codebook entries $\boldsymbol{y}_{p,n}, p = 1,\ 2,\ \ldots,\ P, n = 1,\ 2,\ \ldots,\ N_p$, test the cost effectiveness of a potential split by calculating the decrease in Lagrangian cost

$$\Delta L_{p,n} = \Delta D_{p,n} - \lambda \Delta R_{p,n} \tag{6}$$

where $\Delta D_{p,n}$ is the decrease in distortion and $\Delta R_{p,n}$ is the increase in rate.

Step 3) Sort all entries of the codebooks in decreasing order of $\Delta L_{p,n}$.

Step 4) Starting at the top of the sorted list, codewords are split one by one until a specified criterion is met (N_p is incremented accordingly).

Step 5) Given the new codebooks, we run an M-search encoding of the entire training set.

Step 6) For all stages, and all codebook entries of each stage, update stage codevectors using (5).

Step 7) If target codebook sizes are achieved, stop. Otherwise, go to Step 2.

The initialization in Step 1 consists of setting the single codevector of the first codebook to the centroid of the entire input training set, while the remaining codebooks each contains a single zero codevector. Step 2 evaluates the *reward* of each potential split, $\Delta L_{p,n}$. For a codebook entry $\boldsymbol{y}_{p,n}$, the calculation only involves the corresponding training subset $\boldsymbol{\chi}_{\boldsymbol{p},\boldsymbol{n}} \subset \boldsymbol{X}$. A tentative split of stage codevector $\boldsymbol{y}_{p,n}$ results in two codevectors $\boldsymbol{y}_{p,n'}$ and $\boldsymbol{y}_{p,n''}$. The training subset $\boldsymbol{\chi}_{\boldsymbol{p},\boldsymbol{n}}$ is accordingly subdivided into two subsets, $\boldsymbol{\chi}_{\boldsymbol{p},\boldsymbol{n'}}$ and $\boldsymbol{\chi}_{\boldsymbol{p},\ \boldsymbol{n''}}$ or, equivalently, the corresponding modified training subsets $\phi_{p,n'}$ and $\phi_{p,n''}$. Note that this subdivision is obtained by perturbing the original codevector $\boldsymbol{y}_{p,n}$ into two new vectors, and then optimizing the two new vectors using (5) until convergence.

A split trades an increase in rate for decrease in distortion. We note that a split of $\boldsymbol{y}_{p,n}$, requires each $\boldsymbol{x} \in \boldsymbol{\chi}_{\boldsymbol{p},\boldsymbol{n}}$ to use approximately one extra bit to differentiate the two different paths through the new EC-MSVQ. We thus estimate the rate increase by $\Delta R_{p,n} = |\boldsymbol{\chi}_{\boldsymbol{p},\boldsymbol{n}}|$. The corresponding decrease in distortion is calculated as

$$\Delta D_{p,n} = \sum_{\varphi \in \phi_{\boldsymbol{p},n}} \|\varphi - y_{p,n}\|^2 - \sum_{\varphi \in \phi_{\boldsymbol{p},n'}} \|\varphi - y_{p,n'}\|^2 - \sum_{\varphi \in \phi_{p,n''}} \|\varphi - y_{p,n''}\|^2. \tag{7}$$

Although the rate cost of a split was roughly estimated, the exact design of the *entropy-constrained* MSVQ must be performed while taking into account the probability of quantizer outputs. To reduce complexity, the probability $\Pr(\boldsymbol{i}) = \Pr(i_1,\ i_2,\ \ldots,\ i_P)$ of a path in the EC-MSVQ may be approximated by an mth order Markov model and most commonly by the first-order Markov model ($m = 1$):

$$\Pr(i_1,\ i_2,\ \ldots,\ i_P) \approx \Pr(i_1) \prod_{p=2}^{P} \Pr(i_p | i_{p-1}).$$

At each iteration, and whenever splits occur, all transition probabilities are updated.

The ACL algorithm utilizes the proposed selective splitting approach to EC-MSVQ by performing the selective splitting each time a new training set is generated, and a new quantizer is sought. With the new training set, the current quantizer, $\{Q_1,\ Q_2,\ \ldots,\ Q_P\}$, may exhibit empty cells in some or all of the stages, and the selective splitting algorithm will optimally fill such cells with codevectors that operate at the desired rate-distortion trade-off. As the ACL algorithm converges, the number of empty cells that re-appear with each iteration diminish. On the other hand, as evident from simulations, the CL algorithm may continually require the selective splitting algorithm to fill empty cells.

IV. Simulation Results

The proposed EC-PMSVQ design was tested in the context of low bit rate video coding. We implemented a video codec where 8×8 blocks of residuals are used as vectors. The video sequences are in QCIF format with a frame rate of 10 frames/s. The general structure of the codec is as shown in Fig. 1. The system uses half-pixel motion compensation, and is basically a simplified version of the H.263 scheme where the DCT/quantization module was replaced with the EC-PMSVQ, and each 8×8 block is considered as a separate macroblock. After the first frame, all frames are compressed in interframe mode. Other features of H.263, such as bi-directional prediction and unrestricted motion vectors, can be readily added to the final EC-PMSVQ system. However, we disabled such features in the simulations to obtain a straightforward comparison of DCT with EC-MSVQ. The Lagrange multiplier λ is used to control the rate. In all our simulations, first-order conditional Huffman codes are employed to generate variable length codewords.

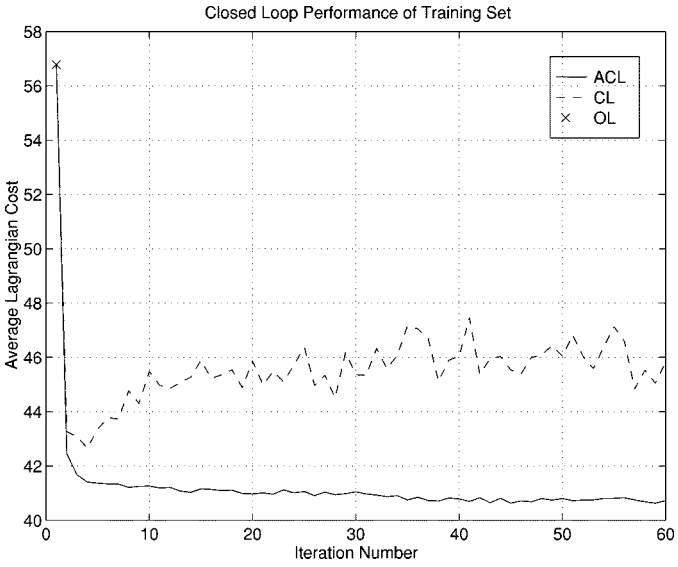

Fig. 5. Performance comparison in terms of average rate-distortion Lagrangian cost on the *thirteen training sequences.* Results are for the EC-PMSVQ available at the end of each iteration. Both techniques are initialized with the outcome of OL design. Note the gradual decrease in Lagrangian cost of the ACL method, and the instability of the CL design procedure.

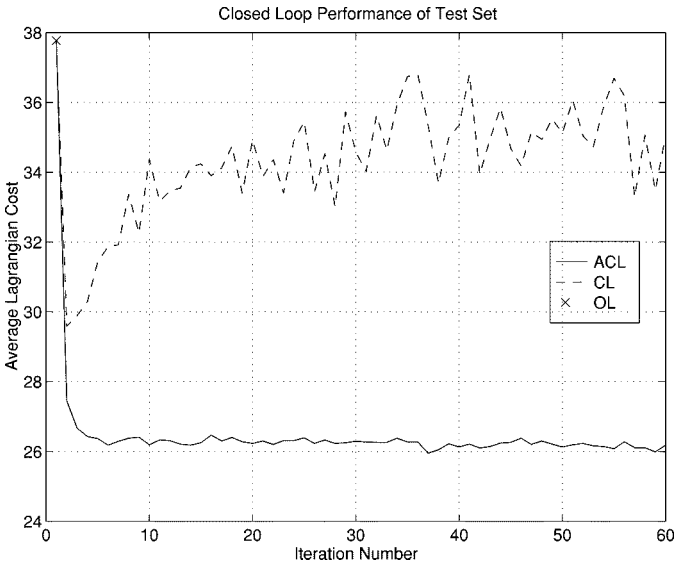

Fig. 6. Performance comparison in terms of average rate-distortion Lagrangian cost on the *three independent test sequences.* Results are for the EC-PMSVQ available at the end of each iteration. Both techniques are initialized with the outcome of OL design. Note the stability and superiority of the ACL approach.

In order for the EC-PMSVQ to be statistically representative, we used a total of 13 video sequences in the training phase. Each video sequence is of length 20 frames. The test set is composed of the three independent (i.e., unused for training) video sequences, namely, "Salesman," "Claire," and "Akiyo," each also of 20 frames. In Figs. 5 and 6, we show the average rate-distortion Lagrangian cost incurred over the *training* sequences and the *test* sequences, respectively, in normal closed-loop operating mode, at the end of each iteration of the design procedure. Note that the ACL algorithm improves with iterations, while the CL algorithm is very unstable. Notice in particular how the CL algorithm becomes unstable right after its first few iterations. These results are in terms of rate-distortion Lagrangian (rather than PSNR) as it allows meaningful averaging over several different input sequences with differing rate requirements.

TABLE I
PERFORMANCE COMPARISON OF H.263 AND THE VARIOUS DESIGNS OF EC-PMSVQ FOR THE TEST IMAGE SEQUENCES "SALESMAN," "CLAIRE," AND "AKIYO." RESULTS FOR EC-PMSVQ SHOWN ARE FOR THE FOLLOWING: 1) OL METHOD, 2) CL METHOD AFTER FIRST ITERATION (HIGHEST TEST PERFORMANCE), 3) CL METHOD AFTER THE COMPLETION OF THE ITERATIVE DESIGN, AND 4) ACL METHOD AFTER COMPLETION OF THE ITERATIONS. THE COMPARISON IS IN TERMS OF PSNR IN DECIBELS, RATE IN Kbits/s, AND THE RATE-DISTORTION LAGRANGIAN COST PER PIXEL

Sequence	Coder	PSNR	Rate	$D+\lambda R$
Salesman	H.263	31.82	19.75	45.07
	OL	30.33	12.65	61.71
	CL (best)	31.61	20.08	47.23
	CL (final)	30.97	33.33	55.97
	ACL	**32.22**	**19.66**	**41.33**
Claire	H.263	36.73	12.96	15.35
	OL	34.99	10.38	21.86
	CL (best)	36.21	16.78	17.54
	CL (final)	35.67	21.17	20.12
	ACL	**36.80**	**12.92**	**15.11**
Akiyo	H.263	34.96	13.80	22.39
	OL	33.54	7.94	29.73
	CL (best)	34.68	15.77	23.99
	CL (final)	33.93	23.95	29.13
	ACL	**35.13**	**13.82**	**21.59**

Table I compares the performance of the various EC-PMSVQ designs. Shown are the average PSNR over each of the individual test sequences, the average bit rates, and the combined average Lagrangian distortion. Bit rates shown are those for coding only the residual. We show the system performance in four settings:

1) OL method;
2) CL method after the first iteration (highest test performance);
3) CL method after the completion of the iterative design;
4) ACL method after completion of the iterative design.

As pointed out earlier, even when the ACL and CL start with same exact initial conditions, the CL method can become very unstable and the end results become uncompetitive. On the other hand, ACL offers stable performance throughout the iterations, and finally achieves a substantial gain over the test sequences. Also, comparing the average rate-distortion Lagrangian $D + \lambda * R$ in Table I shows that improvements were obtained in all test sequences. Here, D is measured in MSE and R in bits/vector, while λ is set to 30 (same value used for training.) For reference, H.263 coding results are shown for bit rates that almost match those of the ACL results. It can be seen that gains of up to 0.4 dB can be achieved.

The EC-PMSVQ design, in this case, involved two quantizers: one quantizer optimized for blocks whose motion vector was zero, and another quantizer optimized for blocks with nonzero motion. Note that the switching information need not be conveyed to the receiver as it is determined by the motion. The parameters used for both quantizers are as follows: $P = 3$ stages, $N = 64$ vectors per stage, and $M = 3$ survivors for the M-search.

V. Conclusions

This correspondence addresses the problem of predictive multistage vector quantizer design for video coding. It extends our recent work on predictive VQ and mitigates longstanding design shortcomings. In particular, it resolves the design stability and empty cell problems.

References

[1] H. Khalil, K. Rose, and S. L. Regunathan, "The asymptotic closed-loop approach to predictive vector quantizer design with applications in video coding," *IEEE Trans. Image Processing*, vol. 10, pp. 15–23, Jan. 2001.

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

[3] F. Kossentini and M. J. T. Smith, "A fast PNN design algorithm for entropy-constrained residual vector quantization," *IEEE Trans. Image Processing*, vol. 7, pp. 1045–1050, July 1998.

[4] F. Kossentini, W. C. Chung, and M. J. T. Smith, "Conditional entropy-constrained residual VQ with application to image coding," *IEEE Trans. Image Processing*, vol. 5, pp. 311–320, Feb. 1996.

[5] K. Rose, H. Khalil, and S. L. Regunathan, "Open-loop design of predictive vector quantizers for video coding," in *Proc. IEEE Int. Conf. Image Processing '98*, vol. 3, Chicago, IL, Oct. 1998, pp. 953–957.

[6] V. Cuperman and A. Gersho, "Vector predictive coding of speech at 16 kbits/s," *IEEE Trans. Commun.*, vol. COM-33, pp. 685–696, July 1985.

[7] Y. Linde, A. Buzo, and R. M. Gray, "An algorithm for vector quantizer design," *IEEE Trans. Commun.*, vol. COM-28, pp. 84–95, Jan. 1980.

[8] P. C. Chang and R. M. Gray, "Gradient algorithms for designing predictive vector quantizers," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-34, pp. 679–690, Aug. 1986.

[9] B. H. Juang and A. H. Gray, "Multiple stage vector quantization for speech coding," in *Proc. IEEE Int. Conf. Acoustics, Speech, Signal Processing*, vol. 1, Apr. 1982, pp. 597–600.

[10] F. Jelinek and J. B. Anderson, "Instrumentable tree encoding of information sources," *IEEE Trans. Inform. Theory*, vol. IT-17, pp. 118–119, Jan. 1971.

[11] C. F. Barnes and R. L. Frost, "Vector quantizers with direct sum codebooks," *IEEE Trans. Inform. Theory*, vol. 39, pp. 565–580, Mar. 1993.

[12] F. Kossentini, M. J. T. Smith, and C. F. Barnes, "Necessary conditions for the optimality of variable-rate residual vector quantizers," *IEEE Trans. Inform. Theory*, vol. 41, pp. 1903–1914, Nov. 1995.

[13] P. A. Chou, T. Lookabaugh, and R. M. Gray, "Entropy constrained vector quantization," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-37, pp. 31–42, Jan. 1989.

[14] W. H. Equitz, "A new vector quantization clustering algorithm," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. 37, pp. 1568–1575, Oct. 1989.

[15] W. A. Finamore, D. P. de Garrido, and W. A. Pearlman, "A clustering algorithm for entropy-constrained vector quantizer design," in *Proc. SPIE 1360, Visual Commun. Image Process. '90*, Lausanne, Switzerland, Nov. 1990, pp. 837–846.

[16] E. A. Riskin and R. M. Gray, "A greedy tree growing algorithm for the design of variable rate vector quantizers," *IEEE Trans. Signal Processing*, vol. 39, pp. 2500–2507, Nov. 1991.

[17] H. Khalil and K. Rose, "Selective splitting approach to entropy-constrained single/multi-stage vector quantizer design," in *Proc. SPIE Conf. Electronic Imaging*, San Jose, CA, Jan. 2000.