Selective Splitting Approach to Entropy-Constrained Single/Multi-Stage Vector Quantizer Design

Hosam Khalil and Kenneth Rose

Department of Electrical and Computer Engineering,
University of California, Santa Barbara, CA 93106

ABSTRACT

A practical tool is proposed to improve the design of various vector quantizer (VQ) structures. Particular emphasis is placed on the design of entropy-constrained VQ (ECVQ), and entropy-constrained multi-stage VQ (EC-MSVQ), whose optimization is notoriously difficult. Traditional design techniques involve an indirect approach where a fixed rate quantizer is gradually modified into a variable rate VQ. We propose a direct design procedure based on selective codevector splitting. The codebooks are grown, using splitting according to a rate-distortion Lagrangian trade-off, to the desired operating point of average bit rate. Extensive simulations in image and video compression are presented and show consistent, significant improvement over standard techniques. For example, in ECVQ design for compression of video residuals, PSNR gains of about 1.0 dB were achieved.

Keywords: Entropy Constrained Vector Quantization, Multi-Stage, Codebook Design

1. INTRODUCTION

Vector quantization is a technique to jointly quantize grouped samples of data. In its most common form, input data is blocked into fixed size vectors prior to quantization. The objective of the design of an effective vector quantizer (VQ) is to identify a set of vectors, called a "codebook", that is representative of the input signal. A representative training set of source vectors is used to facilitate the design procedure. An input vector is matched to the codebook vector that is closest with respect to a predefined distance measure, and the codebook index information can be used to represent the input vector for the purpose of data compression.

Over the years, VQ has matured and more efficient and complex design techniques have emerged for various vector quantizer structures. The need for a particular vector quantizer structure highly depends on the target application. Such applications include speech, image, and video coding, as well as pattern recognition. For example, multi-stage VQ has been widely adopted in several speech coding standards. Variable rate VQ, that comes in structures such as variable dimension VQ and classified VQ, has found application in image and video coding.[1] In this paper, we describe means to improve the design of important VQ techniques such as the (variable rate) entropy-constrained VQ (ECVQ) and the entropy-constrained multi-stage VQ (EC-MSVQ). Other structures such as finite-state VQ and predictive VQ can also benefit from the proposed design.

The organization of the paper is as follows. Section 2 of this paper summarizes some of the best known approaches that are currently used for the design of VQ, ECVQ, and EC-MSVQ. Our proposed selective splitting design technique is developed and applied to the design of ECVQ and EC-MSVQ in Section 3. We present simulation results in Section 4, and conclusions in Section 5.

2. CURRENT DESIGNS

The most popular approach to the design of quantizers is based on an iterative descent algorithm, which was first proposed by Lloyd,[2] and has since been generalized and successfully incorporated into many different frequently used quantizers. In this section, we briefly outline the common approaches to fixed rate VQ, ECVQ, and EC-MSVQ design.

Correspondence: E-mail: hkhalil@scl.ece.ucsb.edu, rose@ece.ucsb.edu

2.1. Fixed Rate Vector Quantizers

The generalized Lloyd algorithm (GLA),[3] an extension of the Lloyd algorithm to the case of vectors, is the most commonly used algorithm for training of fixed rate VQ. The design procedure is an iterative process alternating between two steps. The first step consists of optimally encoding a training set of samples using the current codebook. The second step consists of improving the current codebook by calculating new optimal centroids. These two steps are then iteratively repeated. It is easy to show that, as the steps implement necessary conditions for optimality, the average encoding distortion is monotone non-increasing, and a locally optimal solution is guaranteed.

An important issue in the above design is the initial codebook, as the locally optimal solution depends on the initialization. One solution is to initialize the codebook with a random subset of the training set. Another solution is the splitting algorithm.[3] In the latter case, the algorithm grows large codebooks from small ones: A codebook is first initialized with a single codevector at the centroid of the entire training set. Then, iteratively, all elements of the codebook are split using perturbation causing the codebook to double in size. A re-optimization of the entries is then performed prior to splitting again. The splitting process is no longer needed when the objective codebook size is realized.

Fixed rate VQ is used in applications that require a constant bit rate. Typical applications include speech coding where robustness to channel errors can be better served with a fixed rate bit stream.

2.2. Entropy-Constrained Vector Quantizers

In most applications, individual codebook entries may have different significance. For example, some codevectors may be used more commonly than others. In that case, it may be advantageous to optimize the bit rate based on codevector entropy. This is usually done optimally and jointly with the codebook design itself.

In the case of ECVQ, the standard training technique is due to Chou, Lookabaugh, and Gray,[4] and consists of a modification of the GLA. A fixed rate codebook is used as the initialization. A Lagrangian formulation is then imposed, where the cost of encoding a training vector is a function of both distortion D and encoding rate R (typically estimated by entropy): $L = D + \lambda R$. The Lagrangian multiplier, λ, controls the rate-distortion trade-off and may be used to impose the desired bit rate. The training algorithm starts with a fixed rate ($\lambda = 0$) codebook and modifies it into a variable rate codebook by increasing λ in a series of steps.

ECVQ is used in variable rate applications such as image and video compression. For example, in Ref. 5, ECVQ is found to be very competitive with image standards such as JPEG. Also, recently in video compression,[6] performance better than the H.263 video standard can be achieved.

2.3. Entropy-Constrained Multi-Stage VQ

Multi-stage vector quantization (MSVQ) is widely used in compression of audio signals. MSVQ decomposes the source vector into the sum of codevectors, one per stage (see Fig. 1). Historically, MSVQ was conceived as a sequential quantization operation where each stage simply quantizes the residual of the previous stage, and hence MSVQ is sometimes referred to, in the literature, as residual VQ. More recently, the greedy nature of simple sequential encoding was recognized, and efficient techniques were proposed to seek better approximation of the source vector as combination of stage-vectors.[7] The decoding procedure is very simple and involves direct summing of the individual stage codevectors (see Fig. 2).

Design for MSVQ is more complicated than that of the design of VQ due to interaction between stages in the search for the best codevector. In fact, the search for the representative overall best combination of stage vectors is only optimal with an exhaustive and prohibitively costly search. On the other hand, a simple sequential (stage-by-stage) search can be highly sub-optimal. A close to optimal approach[7] can be used for efficient encoding of input vectors.

Design for entropy constrained MSVQ (EC-MSVQ) is an even more challenging problem. The joint entropy of the VQ stages may be "broken" 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. Joint optimization of the stages where each stage design is performed using the ECVQ design method, was proposed earlier.[8]

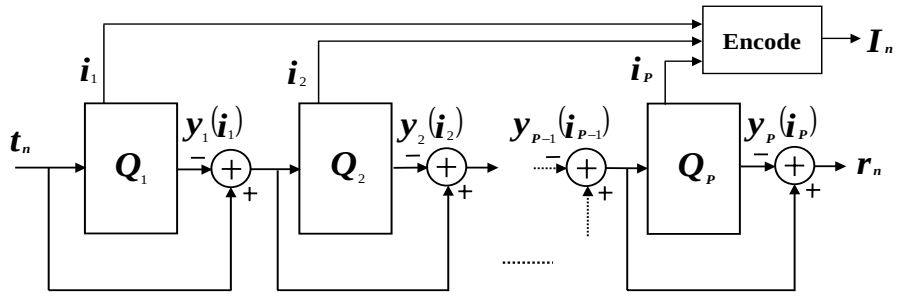


Figure 1. Block diagram of a multi-stage vector quantizer encoder.

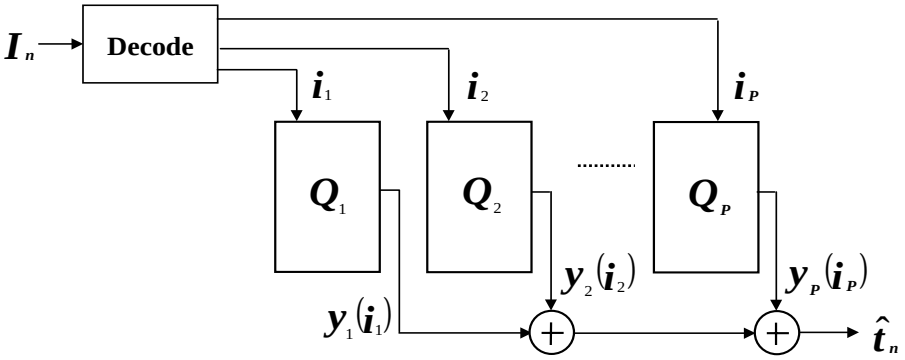


Figure 2. Block diagram of a multi-stage vector quantizer decoder. Note that each stage quantizer Q_p is associated with a corresponding codebook C_p.

2.4. Clustering-based Designs

The pairwise nearest neighbor (PNN) algorithm[9] (also known in pattern recognition as agglomerative hierarchical clustering[10]), was later extended to ECVQ by Finamore et al,[5] and to EC-MSVQ by Kossentini et al.[11] The purpose of the PNN algorithm is to greatly reduce the computational complexity of training algorithms that are variants of GLA.

The PNN algorithm for fixed rate VQ starts by using the training set as the initial codebook. Then the algorithm recursively merges the pair of reproduction vectors, or clusters, that yields the least increase in distortion, until the desired codebook size is reached. The algorithm achieves speed at the expense of performance: when merging a pair of clusters, the effect of the merge on the neighboring clusters is ignored.

In the PNN extension to ECVQ[5] and to EC-MSVQ,[11] pairs are merged when they give the least ratio of increase in distortion for a resulting increase in rate. Objective and subjective performance of PNN-based algorithms is claimed to be only slightly inferior to those based on GLA. Also, in the PNN-based approach to EC-MSVQ design, a traditional fixed rate MSVQ needs to be designed prior to applying the merging procedure of PNN. The details of the actual algorithms are omitted here for space considerations.

3. SELECTIVE SPLITTING ALGORITHM

Though PNN-based algorithms are attractive due to the considerable reductions in design time, it should be noted that such savings are restricted to the design stage and do not improve the actual VQ operational run-time. With the availability of more powerful computers, the design-stage complexity is usually not a crucial issue. We believe that improving the design of VQ is of greater importance.

We propose to use a selective splitting approach, which is the logical reverse of PNN. Selected codewords are split in two, recursively. The use of splitting in the design of a fixed rate VQ was first proposed by Linde, Buzo, and Gray.[3] Since then, many enhancements to the splitting method were proposed.[12] [13] In this paper, we derive splitting as a more general method that can benefit a wider array of VQ structures. We will demonstrate that the proposed methods offer better reproduction than standard GLA-based designs. It should be noted that the selective splitting mechanism is closely related to the greedy splitting approach of Riskin and Gray,[14] where it is used for the design of tree-structured VQ. However, we use splitting as a means to improve the initialization, and not for imposing a structure on the solution. We will denote the GLA-based design of ECVQ and EC-MSVQ by GLA-ECVQ and GLA-EC-MSVQ, respectively, to avoid confusion. The following are the proposed selective splitting algorithms for ECVQ and EC-MSVQ, which we will denote by SS-ECVQ and SS-EC-MSVQ, respectively (Fixed rate VQ is just a special case of SS-ECVQ with λ=0):

3.1. Selective Splitting ECVQ Design

The proposed procedure for ECVQ design is as follows:

Step 1. *Initialize the codebook to a single codevector at the centroid of the entire training set.*

Step 2. *For all entries c_i of the codebook, test for the cost effectiveness of a potential split by calculating*

$$\Delta L_i = \Delta D_i - \lambda \Delta R_i \tag{1}$$

where ΔL_i is the decrease in Lagrangian cost, ΔD_i is the decrease in distortion, and ΔR_i is the increase in rate.

Step 3. *Sort and list all entries of the codebook in decreasing order of ΔL_i.*

Step 4. *Starting at the top of the sorted list, split codewords one by one until a specified criterion is met. Insert the new codewords into the codebook.*

Step 5. *Given the codebook, run GLA-ECVQ over the entire training set.*

Step 6. *If target codebook size is achieved, stop. Otherwise, go to Step 2.*

Comments and Observations:

- In Step 2, we first calculate the distortion D_i of the training subset T_i associated with codevector i. Then we split the codevector into two new vectors, and apply GLA to T_i producing two training subsets $T_i{}'$ and $T_i{}''$. For these two subsets, we calculate the corresponding distortions $D_i{}'$ and $D_i{}''$. We then evaluate $\Delta D_i = D_i - (D_i{}' + D_i{}'')$. If the number of vectors in T_i is N_i, we assume (for simplicity) that we need an extra bit for each vector in T_i resulting in a rate increase of $\Delta R_i = N_i$ bits.
- A "healthy" split is ensured by testing for ΔL_i. Clearly, a negative ΔL_i indicates a counter-productive split, while large positive values of ΔL_i indicate advantageous splits.
- The complexity of testing for a codeword split is not excessive, as the training samples considered are only the subset associated with the codevector. The number of splits per iteration of Step 4 can be either predetermined, or vary depending on the current versus target codebook size. In general, the number of splits per iteration determines the tradeoff between quality and complexity.
- Step 5 allows rectification of "near-sighted" or overly greedy splits of individual codewords, as the whole training set is reconsidered. Usually, a couple of iterations are sufficient to ensure convergence.

3.2. Selective Splitting EC-MSVQ Design

Using notation similar to Ref. 11, it is our objective to design a P-stage EC-MSVQ, where the p^{th} stage quantizer Q_p, $1 \leq p \leq P$, is associated with a stage codebook C_p. Each stage codebook C_p contains N_p stage codevectors $\boldsymbol{y}_p(i_p)$, where $i_p \in I_p$ is the p^{th} stage index and $I_p = \{1, 2, \cdots, N_p\}$. The set of stage mappings $\{Q_1, Q_2, \cdots, Q_P\}$ is equivalent to a single mapping $\boldsymbol{Q}$, which is referred to as the direct-sum vector quantizer. The mapping $\boldsymbol{Q}$ can be described by a direct-sum codebook containing $N = N_1 N_2 \cdots N_P$ direct-sum vectors $\boldsymbol{y}(\boldsymbol{i})$ given by $\boldsymbol{y}(\boldsymbol{i}) = \sum_{p=1}^{P} \boldsymbol{y}_p(i_p)$, where $\boldsymbol{i} = (i_1, i_2, \cdots, i_P) \in \boldsymbol{I}$. Here, the set of all possible indices $\boldsymbol{I} = \{1, 2, \cdots, N\}$ have N corresponding partitions, each defining a direct-sum cell $\boldsymbol{V}(\boldsymbol{i})$.

The common approach to joint multi-stage quantizer design is to view it as a tree structure with an implicit direct-sum codebook. Then, a stage-removed index mapping is defined as $\beta_p(i) = (i_1, i_2, , i_{p-1}, i_{p+1}, \cdots, i_P)$, for $\boldsymbol{i} \in \boldsymbol{I}$, and represents a shortened path through the multi-stage quantizer tree where the p^{th} node has been removed. Each direct-sum codevector $\boldsymbol{y}(\boldsymbol{i})$ can therefore be written as $\boldsymbol{y}(\boldsymbol{i}) = \boldsymbol{g}(\beta_p(\boldsymbol{i})) + \boldsymbol{y}_p(i_p)$, where $\boldsymbol{g}(\beta_p(\boldsymbol{i})) = \sum_{s \neq p}^{P} \boldsymbol{y}_s(i_s)$ is the p^{th} stage-removed direct-sum codevector.

Let $\boldsymbol{X}$ be a k-dimensional random vector described by a probability density function on $\mathcal{R}_k$. For a specific vector $\boldsymbol{x}$ that is encoded by index $\boldsymbol{i}$, we define the p^{th} stage-removed residual vector by $\boldsymbol{\gamma}_p = \boldsymbol{x} - \boldsymbol{g}(\beta_p(\boldsymbol{i}))$. Also, define $\mathcal{I}_p(i_p) \subset \boldsymbol{I}$ to be the subset of indices where i_p is the p^{th} element of $\boldsymbol{i}$. The i_p^{th} stage-removed residual set $\mathcal{V}_p(i_p) \subset \mathcal{R}_k$ is then defined as

$$\mathcal{V}_p(i_p) = \bigcup_{\boldsymbol{i} \in \mathcal{I}_p(i_p)} (\boldsymbol{V}(\boldsymbol{i}) - \boldsymbol{g}(\beta_p(\boldsymbol{i}))) \tag{2}$$

where $\mathcal{V}_p(i_p)$ is the set of all stage-removed residual vectors formed from input vectors that map to one of the set of direct-sum codevectors containing $\boldsymbol{y}_p(i_p)$ in their construction. Now each codevector in any stage of the EC-MSVQ will be designed such that it is a centroid of its respective stage-removed residual set $\mathcal{V}_p(i_p)$.

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

Step 1. Initialize the P codebooks, each to contain a single vector.

Step 2. For all entries $y_p(i_p)$ in all codebooks $1 \leq p \leq P$, test for the cost effectiveness of a potential split by calculating

$$\Delta L_{p,i_p} = \Delta D_{p,i_p} - \lambda \Delta R_{p,i_p} \tag{3}$$

where $\Delta L_{p,i_p}$ is the decrease in Lagrangian cost, $\Delta D_{p,i_p}$ is the decrease in distortion, and $\Delta R_{p,i_p}$ is the increase in rate.

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

Step 4. Starting at the top of the sorted list, codewords are split one by one until a specified criterion is met, and the new codewords are inserted into their corresponding codebooks.

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

Step 6. For all stages, $1 \leq p \leq P$, and all codebook entries of stage p, calculate new centroids of stage-removed sets defined by Equation 2.

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

Comments and Observations:

- For the initialization of Step 1, the single codevector of the first codebook contains the centroid of the entire input training set, while the remaining codebooks each contains a single zero-valued vector.

- In Step 2, we calculate $\Delta L_{p,i_p}$ to evaluate the *reward* of a split. For a particular stage codebook entry $y_p(i_p)$, the set of input training vectors $\boldsymbol{V}(\boldsymbol{i})$ (there are N_{p,i_p} such vectors) that use $y_p(i_p)$ as part of their construction will be considered separately in what follows: Say in stage p, the codebook entry having index i_p and its corresponding direct-sum stage codevector $y_p(i_p)$ is split into two codebook entries having indices $i_p^{'}$ and $i_p^{''}$, and corresponding stage codevectors $y_p(i_p^{'})$ and $y_p(i_p^{''})$, respectively. The set of stage-removed vectors $\mathcal{V}_p(i_p)$ will be divided, accordingly, into two subsets, $\mathcal{V}_p(i_p^{'})$ and $\mathcal{V}_p(i_p^{''})$, whose centroids become $y_p(i_p^{'})$ and $y_p(i_p^{''})$, respectively. This process involves perturbing the vector $y_p(i_p)$ into two new vectors, and then optimizing the two new vectors using GLA over $\mathcal{V}_p(i_p)$. After convergence, the two resulting vectors will be centroids of the two subsets $\mathcal{V}_p(i_p^{'})$ and $\mathcal{V}_p(i_p^{''})$, respectively, where

$$\mathcal{V}_p(i_p) = \{\mathcal{V}_p(i_p^{'}), \mathcal{V}_p(i_p^{''})\} \tag{4}$$

To decide if this is an advantageous split, we need to calculate the price we incur in rate. For this particular stage codevector ($y_p(i_p)$), each of the input training vectors in $\boldsymbol{V}(\boldsymbol{i})$ that use $y_p(i_p)$ in their construction (i.e., the elements in $\mathcal{I}_p(i_p)$) will require approximately one extra bit to differentiate the two different paths through the new EC-MSVQ. Thus a total of N_{p,i_p} extra bits will be needed, resulting in $\Delta R_{p,i_p} = N_{p,i_p}$.

For each candidate of splitting, we calculate $\Delta D_{p,i_p}$ where

$$\Delta D_{p,i_p} = \Delta D(p, i_p, i_p^{'}, i_p^{''}) = \sum_{\gamma_p \in \mathcal{V}_p(i_p)} d(\gamma_p, y_p(i_p)) - \sum_{\gamma_p \in \mathcal{V}_p(i_p^{'})} d(\gamma_p, y_p(i_p^{'})) - \sum_{\gamma_p \in \mathcal{V}_p(i_p^{''})} d(\gamma_p, y_p(i_p^{''})) \tag{5}$$

Then, we evaluate

$$\Delta L_{p,i_p} = \Delta D_{p,i_p} - \lambda N_{p,i_p} \tag{6}$$

- A "healthy" split is ensured by testing for $\Delta L_{p,i_p}$. Clearly, a negative $\Delta L_{p,i_p}$ indicates a counter-productive split, while large positive values of $\Delta L_{p,i_p}$ indicate advantageous splits.

- The complexity of testing for a codeword split is not excessive, as the training samples considered are only the subset associated with the codevector. The number of splits per iteration of Step 4 can be either predetermined, or vary depending on the current versus target codebook size. In general, the number of splits per iteration determines the tradeoff between quality and complexity.

- Since the design is for an *entropy-constrained* MSVQ, an M-search features a search for minimum Lagrangian cost rather than minimum distortion. See Ref. 8 for conditions of optimality. The probability $pr(\boldsymbol{i}) = pr(i_1, i_2, \cdots, i_P)$ of a path in the EC-MSVQ is usually approximated using an m-th order Markov model where

$$pr(i_1, i_2, \cdots, i_P) \approx pr(i_P \mid i_{P-1}, \cdots, i_{P-m}).pr(i_{P-1} \mid i_{P-2}, \cdots, i_{P-m}) \cdots .pr(i_2 \mid i_1)pr(i_1) \tag{7}$$

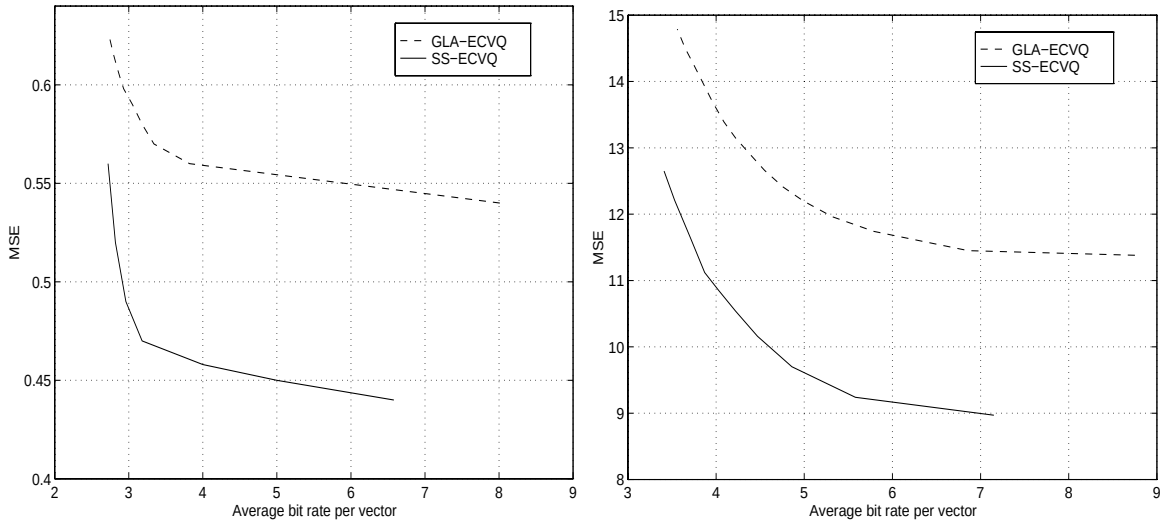
which reduces for single-stage conditioning ($m = 1$), as is commonly used, to

$$pr(i_1, i_2, \cdots, i_P) \approx pr(i_P \mid i_{P-1}).pr(i_{P-1} \mid i_{P-2}) \cdots .pr(i_2 \mid i_1)pr(i_1) \tag{8}$$

Whenever a stage codebook vector is split in Step 4, the necessary stage conditional probabilities relating to the split node will need to be updated. We update all probabilities after splits and with each new iteration.

- The codeword for any path through the EC-MSVQ then becomes a concatenation of stage codewords. Stage codeword-lengths can be estimated using conditional entropy over the training set, or actual bit rates based on conditional Huffman tables can be used for more accurate estimates.

- For the M-search, best encoding indices are searched in stage order. After we find the best M candidates from the first stage, we search for the best M candidates that can emanate from the previous M candidates. At stage p, we calculate the Lagrangian L which is used to specify the total cost of encoding an input vector *up until* stage p such that the bit rate used is that of the concatenated stage codewords that are needed up until stage p. For the results shown in this paper, conditional Huffman codes are used for accurate performance estimates.



(a) (b)

Figure 3. Rate-distortion comparison of traditional GLA-ECVQ and the proposed SS-ECVQ design for (a)"Carphone" sequence, and (b) "Claire" sequence.

- Steps 5 and 6 allow to rectify "near-sighted" or overly greedy splits of individual codewords, as the whole training set is reconsidered. Usually, a couple of iterations are sufficient to ensure convergence.

4. RESULTS

In this section, we demonstrate the results of the proposed design algorithms. The tests for evaluating the selective splitting algorithm are divided into two experiments. The first experiment evaluates the new single-stage ECVQ Design, while the second test evaluates the new EC-MSVQ Design.

We use video data for testing single-stage ECVQ. The training data consists of half-pixel motion compensated residuals of a 30 frame portion of each of the "Carphone" and "Claire" QCIF sequences. Vectors of size 8×8 and a target codebook size of 2048 vectors are used. Figure 1 shows a comparison for the average distortion per pixel at different bit rates. We compare the traditional GLA-ECVQ, where λ is gradually increased to generate several points on the curve, with that of the proposed SS-ECVQ, where we apply the selective splitting algorithm in which a new codebook is designed for each λ. The two video sequences were particularly chosen to show performance in case of high-motion (Carphone) and low-motion (Claire). It can be seen that a significant gain is achieved by the new algorithm. Specifically, gains of about 1dB in PSNR can be achieved in both cases. The coder was tested on the training set (instead of a test set) in this case, to clarify the results, as training of predictive vector quantizers for video is a complicated procedure, and the closed loop operation of such coders would result in an unclear comparison. It should also be noted that in the SS-ECVQ method, a new codebook is designed directly without the need for the gradual design of GLA-ECVQ.

In the second experiment, an entropy constrained multi-stage vector quantizer is trained using our selective splitting approach (SS-EC-MSVQ). Results are compared to the traditional design approach of GLA-EC-MSVQ. The training data consists of a group of six images, and testing is performed on two independent images (not in the training set). The images are divided into blocks of 4×4 pixels and directly applied to the EC-MSVQ. The image coder is simple to focus the results on the vector quantizer issues. A 3-stage EC-MSVQ, with each stage of size 32 codevectors is designed. For the M-search, three candidates are used. For the entropy calculations, single stage conditional probabilities are used. Results in PSNR at different bit rates are shown in Fig. 4 for "Lena" and "Goldhill" images. It can be seen that a gain of up to 0.25 dB can be achieved on an independent test set. An important advantage of the algorithm is that design of the codebook is direct, in the sense that a codebook is built directly to operate at the given entropy constrained operational point, rather than gradually imposing the entropy starting with a fixed-rate VQ.

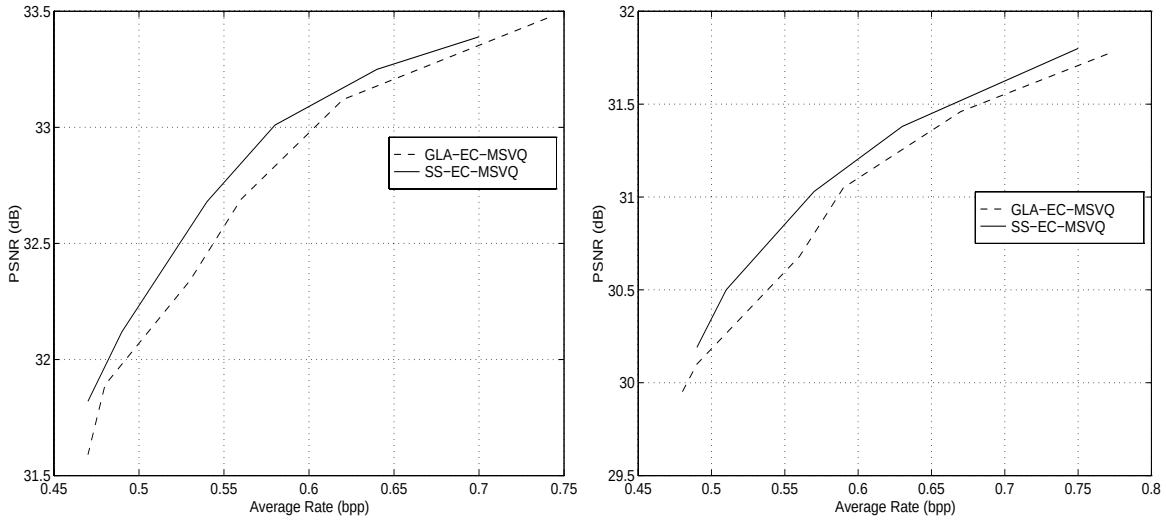


Figure 4. Rate-PSNR comparison of traditional GLA-ECVQ and the proposed SS-ECVQ design for (a)"Lena" image, and (b) "Goldhill" image.

REFERENCES

1. A. Gersho and R, M. Gray, *Vector Quantization and Signal Compression*. Boston, MA: Kluwer, 1992.
2. S. P. Lloyd, "Least squares quantization in PCM," *IEEE Trans. Inform. Theory.*, vol. IT-28, pp. 129-136, Mar. 1982, (Previously an unpublished Bell Lab Tech. Note, 1957.)
3. Y. Linde, A. Buzo, and R. M. Gray, "An algorithm for vector quantizer design," *IEEE Trans. Communications*, vol. COM-28, pp. 84-95, Jan. 1980.
4. P. A. Chou, T. Lookabaugh, and R. M. Gray, "Entropy-constrained vector quantization," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. 37, no. 1, pp. 31-42, Jan. 1989.
5. W. A. Finamore, D. P. de Garrido, and W. A. Pearlman, "A clustering algorithm for entropy-constrained vector quantizer design," *Proc. SPIE 1360, Visual Commun. Image Process.* '90, Lausanne, Switzerland, pp. 837-846, Nov. 1990.
6. K. Rose, H. Khalil, and S. L. Regunathan, "Open-loop design of predictive vector quantizers for video coding," *Proc. IEEE Intl. Conf. on Image Processing*, ICIP'98, Chicago, IL, Oct. 1998.
7. W. P. LeBlanc, B. Bhattacharya, S. A. Mahmoud, and V. Cuperman, "Efficient search and design procedures for robust multi-stage VQ of LPC parameters for 4 kb/s speech coding," *IEEE Trans. Speech and Audio Proc.* vol. 1, no. 4, pp. 373-385, Oct. 1993.
8. F. Kossentini, M. J. T. Smith, and C. F. Barnes, "Necessary conditions for the optimality of variable-rate residual vector quantizers," *IEEE Trans, Information Theory*, vol. 41, no. 6, pp. 1903-1914, Nov. 1995.
9. W. H. Equitz, "A new vector quantization clustering algorithm," *IEEE Trans. Acoust., Speech, Signal Proc.*, vol. 37, no. 10, pp. 1568-1575, Oct. 1989.
10. R. O. Duda and P. E. Hart, Pattern Classification and Scene Analysis. John Wiley & Sons, Inc., 1973.
11. F. Kossentini and M. J. T. Smith, "A fast PNN design algorithm for entropy-constrained residual vector quantization," *IEEE Trans. Image Processing*, vol. 7, no. 7, pp. 1045-1050, July 1998.
12. P. Franti, T. Kaukoranta, and O. Nevalainen, "On the splitting method for vector quantization codebook generation," *Opt. Eng.*, vol. 36, pp.3043-51, Nov. 1997.
13. T. Kaukoranta, P. Franti, and O. Nevalainen, "A new iterative algorithm for VQ codebook generation," *Proc. IEEE Intl. Conf. on Image Processing*, ICIP'98, Chicago, IL, vol. 2, pp. 589-93, Oct. 1998.
14. 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, no. 11, pp. 2500-2507, Nov. 1991.