ON NONLINEAR UTILIZATION OF INTERVECTOR DEPENDENCY IN VECTOR QUANTIZATION

Mikael Skoglund

Signal Processing, Dept. of Signals, Sensors and Systems
Royal Institute of Technology, S-100 44 Stockholm, Sweden
Email - skoglund@s3.kth.se

Jan Skoglund

Dept. of Information Theory
Chalmers University of Technology, S-412 96 Göteborg, Sweden
Email - jans@it.chalmers.se

ABSTRACT

This paper presents an approach to vector quantization of sources exhibiting intervector dependency. We present the optimal decoder based on a collection of received indices. We also present the optimal encoder for such decoding. The optimal decoder can be implemented as a table look-up decoder, however the size of the decoder codebook grows very fast with the size of the collection of utilized indices. This leads us to introduce a method for storing an approximation to the set of optimal decoder vectors, based on linear mapping of a block code vector quantization. In this approach a heavily reduced set of parameters is employed to represent the codebook. Furthermore, we illustrate that the proposed scheme has an interpretation as nonlinear predictive quantization. Numerical results indicate high gain over memoryless coding and memory quantization based on linear predictive coding. The results also show that the sub-optimal approach performs close to the optimal.

1. INTRODUCTION

In modern applications of vector quantization, e.g. in speech coding, one of the main goals is attaining high quality at low bit rate. A common technique for enhancing performance at a particular rate is to utilize the memory in the source signal in the coding process. In this paper we emphasize the utilization of memory in the transmitted *index* process $\{I_n\}$ resulting from the quantization. The memory in the index process is assumed to stem from intervector dependency in the process $\{\mathbf{X}_n\}$ of source vectors. To illustrate the principle of the proposed method, consider the decoding problem. Given that, at time n, the decoder makes the observation $I_n = i_n$ an ordinary, memoryless, VQ decoder would produce the centroid $\mathbf{c}(i_n) = E[\mathbf{X}_n|I_n = i_n]$ as estimate of the transmitted source vector $\mathbf{X}_n$. However, in many practical situations we can assume that (*i*); the index process $\{I_n\}$ contains memory that can be utilized by the decoder for enhanced performance, and (*ii*); the decoder knows about the collection of a number of, say, M indices received earlier, that is, $i_{n-1}, \ldots, i_{n-M}$. Since the index process is assumed to have memory, the decoder should thus produce as estimate the "multi-index" centroid $E[\mathbf{X}_n|i_n, \ldots, i_{n-M}]$ instead of the "single-index" centroid $\mathbf{c}(i_n)$ for decoding, at time n. This idea, which is made more precise below, is central to this paper. Similar conclusions regarding the use of multi-centroid decoding have been drawn before in, e.g., [1] for vector quantization and in [2] for scalar encoding. See also [3] for a general treatment of nonlinear estimation using vector quantization.

Decoding based on $E[\mathbf{X}_n|i_n, \ldots, i_{n-M}]$ can be implemented as a table look-up. However, the size of the decoder codebook generally grows extremely fast as a function of M. Because of this drawback of the proposed method, we in this paper put considerable effort in introducing a method for sub-optimal decoding based on linear mapping of a block code VQ (LMBC VQ; see [4, 5]) for *storing* the multi-index codebook at the decoder. In LMBC VQ the number of degrees of freedom in representing the codebook is constrained, and hence a smaller set of parameters has to be stored in order to represent the codebook. Another contribution of the paper is the consideration of optimal encoding for decoding based on the collection of received indices and results for joint encoder-decoder design.

2. PRELIMINARIES

To begin describing the problem under consideration carefully, consider a d-dimensional stationary vector process $\{\mathbf{X}_n\}$, where $\mathbf{X}_n \in \mathbb{R}^d$, and a vector quantizer with encoder mapping ε and with decoder δ. At time n the encoder produces an index $I_n \in \mathcal{I}_N$, where $\mathcal{I}_N \triangleq \{0, 1, \ldots, N-1\}$, and where we assume that $N = 2^L$ (hence, the rate of the code is $R = L/d$), and the decoder computes a source vector estimate $\hat{\mathbf{X}}_n$.

As described in the introduction, the decoder is assumed to be a function of a collection of observed indices. More precisely, in denoting by $\mathbf{I}_n \in \mathcal{I}_N^{M+1}$ the collection of the the present and the earlier M transmitted indices, that is $\mathbf{I}_n = (I_n, \ldots, I_{n-M})$, then when the observation $\mathbf{I}_n = \mathbf{i}_n$ is made, the decoder δ is a function of $\mathbf{i}_n$, according to

$$\hat{\mathbf{X}}_n = \delta(\mathbf{i}_n). \tag{1}$$

In an analogous manner the memory in the index process is utilized in the encoding: Denote as $\mathbf{J}_n \in \mathcal{I}_N^M$ the earlier indices alone, that is $\mathbf{J}_n = (I_{n-1}, \ldots, I_{n-M})$, then, given that $\mathbf{J}_n = \mathbf{j}_n$ and that $\mathbf{X}_n = \mathbf{x}_n$ the encoder is a function,

$$i_n = \varepsilon(\mathbf{x}_n, \mathbf{j}_n) \tag{2}$$

of the source vector, $\mathbf{x}_n$, and of the earlier transmitted indices $\mathbf{j}_n$.

To measure system performance, we throughout employ the mean squared error (MSE) distortion measure

$$D = E\|\mathbf{X}_n - \hat{\mathbf{X}}_n\|^2, \tag{3}$$

and "optimal" will throughout refer to the minimization of D. Next section is devoted to optimal encoding and decoding.

3. OPTIMAL ENCODING AND DECODING

Here we treat optimal decoding, for a given encoder, and optimal encoding, for a given decoder. We start by investigating the decoding.

3.1. Optimal decoding

For a given encoder, and a given source, it is straightforward to realize that the optimal decoder is the minimum mean squared error (MMSE) estimator (c.f., [1] and [2])

$$\hat{\mathbf{X}}_n(\mathbf{i}) = E[\mathbf{X}_n|\mathbf{I}_n = \mathbf{i}]. \tag{4}$$

Given $\mathbf{I}_n = \mathbf{i}$, the decoder $\hat{\mathbf{X}}_n(\mathbf{i})$ is hence a function of the $M+1$ integers in $\mathbf{i}$. Denote, for simplicity, the N^{M+1} possible vectors as $\mathbf{v}(\mathbf{i})$. The decoder can then be implemented as a table look-up, by storing the vectors $\{\mathbf{v}(\mathbf{i})\}$. However, the size of the decoder codebook $\{\mathbf{v}(\mathbf{i})\}$ grows exponentially according to $2^{L(M+1)}$ with the resolution L and with the size of the memory M. In Section 4 we will therefore describe a method for storing a reduced-size approximation to $\{\mathbf{v}(\mathbf{i})\}$ based on LMBC VQ. As a prerequisite to Section 4, the following is a description of the Hadamard transform representation for VQ codebooks [6].

The set $\{\mathbf{v}(\mathbf{i})\}$ of possible decoder vectors can be represented as $\mathbf{v}(\mathbf{i}) = \mathbf{T}\mathbf{h}(\mathbf{i})$ where $\mathbf{T}$ is a fixed transform matrix and the vector $\mathbf{h}(\mathbf{i})$ is defined as (c.f. [6])

$$\mathbf{h}(\mathbf{i}_n) = \tilde{\mathbf{h}}(i_{n-M}) \otimes \cdots \otimes \tilde{\mathbf{h}}(i_{n-1}) \otimes \tilde{\mathbf{h}}(i_n), \tag{5}$$

where "$\otimes$" denotes the Kronecker matrix product and where

$$\tilde{\mathbf{h}}(i) = \begin{bmatrix} 1 \\ b_L(i) \end{bmatrix} \otimes \cdots \otimes \begin{bmatrix} 1 \\ b_2(i) \end{bmatrix} \otimes \begin{bmatrix} 1 \\ b_1(i) \end{bmatrix}. \tag{6}$$

Here, for some (arbitrary) integer $i \in \mathcal{I}_N$, the bits (in a $\{\pm 1\}$ representation) of the integer are denotes as $b_l(i) \in \{\pm 1\}$, $l = 1, \ldots, L$. Consequently, the vector $\mathbf{h}(\mathbf{i})$ contains all possible products of the different bits of the indices contained in $\mathbf{i}$. More precisely, in interpreting $\mathbf{i}$ as the integer $K(\mathbf{i}) \triangleq \sum_{m=0}^{M} i_{n-m} N^m$, $\mathbf{h}(\mathbf{i})$ is the Kth column of an N^{M+1} by N^{M+1} Hadamard matrix $\mathbf{H}$. In the following, we will use both the vector notation $\mathbf{i}$ and the integer interpretation $K(\mathbf{i})$ interchangeably. The mapping matrix $\mathbf{T}$ is defined uniquely by the set of codebook vectors and is obtained as the Hadamard transform $\mathbf{T} = N^{-(M+1)}\mathbf{V}\mathbf{H}$ where $\mathbf{V}$ contains all the codebook vectors as columns. The principle of using the Hadamard transform for representing (memoryless) VQ codebooks, is described more thoroughly in [6].

3.2. Optimal encoding

Having presented the optimal decoder, for fixed encoding, we continue here with a treatment of optimal encoding. The optimal encoder transmits the value of i_n that minimizes the expected value of $\|\mathbf{X}_n - \hat{\mathbf{X}}_n\|^2$ conditioned that $\mathbf{J}_n = \mathbf{j}_n$, $I_n = i_n$ and $\mathbf{X}_n = \mathbf{x}_n$. That is,

$$\varepsilon(\mathbf{x}_n, \mathbf{j}_n) = \arg \min_{i_n \in \mathcal{I}_N} \|\mathbf{x}_n - \hat{\mathbf{X}}_n(i_n, \mathbf{j}_n)\|^2. \tag{7}$$

According to the commonly used principle in VQ design [7], one straightforward way of designing an encoder-decoder pair is as follows: (0) Chose an initial encoder; (*i*) determine the optimal decoder for the given encoder; (*ii*) determine the new optimal encoder for the new decoder; (*iii*) repeat from (*i*). Codes designed according to this principle are evaluated in Section 5.

3.3. Interpretation as nonlinear prediction

An interesting interpretation of the proposed method based on (7) and the Hadamard representation, $\mathbf{T}\mathbf{h}(\mathbf{i})$, for the decoder vectors, can be obtained as follows: Note that according to (5) and (6), the vector $\mathbf{h}(\mathbf{i})$ can be subdivided into three parts, as

$$\mathbf{h}(\mathbf{i}) = \mathbf{h}_{\text{new}} + \mathbf{h}_{\text{mixed}} + \mathbf{h}_{\text{old}}, \tag{8}$$

where $\mathbf{h}_{\text{new}}$ contains the possible products of the bits, $b_l(i_n)$, of the index i_n (and zeros in other positions), $\mathbf{h}_{\text{mixed}}$ contains mixed terms, that is, products of bits of i_n and bits of old indices $\mathbf{j}_n$, and $\mathbf{h}_{\text{old}}$ contains products of old bits only (terms depending only on the indices of $\mathbf{j}_n$). Thus, the decoder vector $\mathbf{v}(\mathbf{i}_n)$ can be written

$$\mathbf{v}(\mathbf{i}_n) = \mathbf{T}\mathbf{h}_{\text{new}}(i_n) + \mathbf{T}\mathbf{h}_{\text{mixed}}(i_n, \mathbf{j}_n) + \mathbf{T}\mathbf{h}_{\text{old}}(\mathbf{j}_n). \tag{9}$$

Consequently, the optimal encoder (7) can be reformulated as

$$\varepsilon(\mathbf{x}_n, \mathbf{j}_n) = \arg \min_{i_n \in \mathcal{I}_N} \|\mathbf{e}_n - \mathbf{T}\mathbf{h}_{\text{new}}(i_n) - \mathbf{T}\mathbf{h}_{\text{mixed}}(i_n, \mathbf{j}_n)\|^2, \tag{10}$$

where $\mathbf{e}_n = \mathbf{x}_n - \mathbf{T}\mathbf{h}_{\text{old}}(\mathbf{j}_n)$. The vector $\mathbf{e}_n$ can be interpreted as a *prediction error*, since it is formed as the difference between the present vector $\mathbf{x}_n$, to be coded, and a prediction $\mathbf{T}\mathbf{h}_{\text{old}}(\mathbf{j}_n)$ based on the previously transmitted indices $\mathbf{j}_n$. Hence we see that the encoder tries to represent the prediction error $\mathbf{e}_n$ as closely as possible using a "new" codevector $\mathbf{T}\mathbf{h}_{\text{new}}(i_n)$ *and* a codevector $\mathbf{T}\mathbf{h}_{\text{mixed}}(i_n, \mathbf{j}_n)$ that depends, in a nonlinear fashion, on *both* the new index i_n (to be transmitted) and the old indices $\mathbf{j}_n$. Consequently, the proposed method has an interpretation as nonlinear predictive coding. Assuming that the encoder-decoder pair has been trained for a particular source, the matrix $\mathbf{T}$ is determined by the resulting decoder codebook. In the interpretation as predictive coding we note that this matrix determines both the VQ (the codebook) *and* the "predictor". Hence, the design gives a *joint* optimization of the VQ and the (nonlinear) predictor. Other work on nonlinear predictive coding in VQ for speech coding has been presented in, e.g., [8] (see also [3]). In passing we note that another, perhaps more obvious, interpretation of the proposed method is an interpretation as finite-state VQ [7], where the "state" is determined by the old indices collected in $\mathbf{j}_n$.

4. LMBC-BASED DECODING AND ENCODING

The optimal decoder can, as commented on above, be implemented by storing the entire set of decoder vectors $\{\mathbf{v}(\mathbf{i})\}$ at the decoder. However, the size of the decoder codebook grows exponentially with the memory M, so the decoder quickly becomes intractable in terms of storage requirements. In this paper we emphasize the use of LMBC VQ [5, 4] for storing an approximation to the decoder vectors. This approach is based on the representation $\mathbf{v}(\mathbf{i}) = \mathbf{T}\mathbf{h}(\mathbf{i})$. From this expression it is evident that the decoder can equivalently be implemented by storing the mapping matrix $\mathbf{T}$. In LMBC the Hadamard representation is *constrained* so that only a (relatively small) number of the columns of $\mathbf{T}$ are allowed to be non-zero. This is equivalent to using a representation $\mathbf{G}\mathbf{g}(\mathbf{i})$, where the vector $\mathbf{g}$ is defined from a sub-set of the components of the "full-freedom" Hadamard column $\mathbf{h}(\mathbf{i})$. To be more specific, we illustrate the principle using the, so called, E3 code. As described in [4, 5] the E3 code, for $\mathbf{g}(\mathbf{i})$, is defined to contain $+1$ in the first position, all linear terms (i.e., the bits of $K(\mathbf{i})$) and

all nonlinear terms (products of bits) of *weight three* (all possible products of three different bits among the $(M+1)L$ bits of $K(\mathbf{i})$). Consequently, in the E3 code the LMBC codevector $\mathbf{g}$ is of size $1 + L(M+1) + (L(M+1))!/(6(L(M+1)-3)!)$. We refer to [4, 5] for a comprehensive description of the LMBC VQ approach (with application to the design of memoryless vector quantizers). These references also give examples of other LMBC codes and some guidelines for choosing a particular code. In Section 5 we present results for both "full-freedom" optimal decoding and LMBC-based decoding, using the described E3 code. (For LMBC we have chosen to employ the E3 code in our results.)

4.1. Decoding

The LMBC-based decoder stores the mapping matrix $\mathbf{G}$ and calculates, for each time-instant n, the vector $\mathbf{g}(\mathbf{i}_n)$ to produce the decoded vector $\mathbf{G}\mathbf{g}(\mathbf{i}_n)$. The dependence of $\mathbf{g}(\mathbf{i}_n)$ on $\mathbf{i}_n$ is determined by the employed LMBC code. Since the product $\mathbf{G}\mathbf{g}(\mathbf{i}_n)$ has to be computed for each time-instant, the price for the reduced storage capacity requirement can be observed to be an increased computational complexity in the decoding (compared to table look-up decoding which is of virtually zero computational complexity). However, the decoding complexity can generally be assumed to be moderate since, as mentioned, the number of terms of the vector $\mathbf{g}(\mathbf{i}_n)$ is relatively low and hence the complexity of evaluating the matrix product $\mathbf{G}\mathbf{g}(\mathbf{i}_n)$ is low.

When the vector $\mathbf{g}$ is unconstrained, that is, when $\mathbf{g}(\mathbf{i}) = \mathbf{h}(\mathbf{i})$, the optimal decoder matrix is $\mathbf{G} = \mathbf{T}$, where $\mathbf{T}$ is the Hadamard transform of the set of optimal decoder vectors $\{\mathbf{v}(\mathbf{i})\}$ (c.f. Section 3.1). However, when constraining $\mathbf{g}$ according to the LMBC principle the codebook is defined from the mapping matrix $\mathbf{G}$ which has to be optimized for the given encoder. More precisely, it is straightforward to show that in the MMSE sense the matrix $\mathbf{G}$ should be chosen according to (c.f., [4, 5])

$$\mathbf{G} = E[\mathbf{X}_n \mathbf{g}^T]\{E[\mathbf{g}\mathbf{g}^T]\}^{-1}, \tag{11}$$

where $\mathbf{g} = \mathbf{g}(\mathbf{I}_n)$.

4.2. Encoding

The optimal encoder, for a given decoder $\mathbf{G}\mathbf{g}(\mathbf{i}_n)$, is

$$\varepsilon(\mathbf{x}_n, \mathbf{j}_n) = \arg \min_{i_n \in \mathcal{I}_N} \|\mathbf{x}_n - \mathbf{G}\mathbf{g}(i_n, \mathbf{j}_n)\|^2. \tag{12}$$

An LMBC-based encoder-decoder pair can now be designed based on; (0) an initial choice of encoder; (*i*) the selection of a particular LMBC code [4, 5] in order to define $\mathbf{g}(\mathbf{i})$; (*ii*) the expression (11) defining the mapping matrix $\mathbf{G}$, and; (*iii*) the expression (12) for the optimal encoder, given a decoder. This procedure can then be iterated from (*ii*) to produce a final encoder-decoder pair.

Note that the computation of $\varepsilon(\mathbf{x}_n, \mathbf{j}_n)$ requires a search over $\mathcal{I}_N$, where for each value of the new index, i_n, to be tested, the vector $\mathbf{G}\mathbf{g}(i_n, \mathbf{j}_n)$ is calculated and compared to $\mathbf{x}_n$. The number of such vectors to be evaluated is N. The complexity of the "brute-force" method of searching by evaluating first $\mathbf{g}(i_n, \mathbf{j}_n)$, based on the description of the employed LMBC code, and then the matrix product $\mathbf{G}\mathbf{g}$, can be quite high (depending on which code is employed). Most LMBC codes [4, 5], however, have sufficient amount of structure to allow for considerable simplifications in the search. To illustrate this, given $\mathbf{j}_n$ let i be the value of i_n to be tested, then $\mathbf{g} = \mathbf{g}(i, \mathbf{j}_n)$ and $\mathbf{G}\mathbf{g}$ are to be evaluated for all $i \in \mathcal{I}_N$. Now assume that the possible values for i are tested in Gray code order; $i = G(0), G(1), \ldots, G(N-1)$, where $G(k)$ defines the Gray encoding of the integer k (that is, $G(k) = k \odot (k/2)$, where "$\odot$" denotes bit-wise modulo-2 addition), then for each index to test it is known that only *one* bit has been altered compared to the previous index that was tested. Since an alteration in one bit always influences an equal number of positions[1] in the vector $\mathbf{g}$, it is straightforward to store a table listing the numbers of the elements of $\mathbf{g}$ that change sign when a new index is tested. Then if $\tilde{\mathbf{X}}$ is the value of $\mathbf{G}\mathbf{g}$ tested against $\mathbf{x}_n$ in the previous comparison, the new vector to test is obtained by adding to $\tilde{\mathbf{X}}$ (or subtracting, depending on the previous signs) two times a sum over the columns of $\mathbf{G}$ that correspond to positions in which $\mathbf{g}$ has changed sign. Using this technique, the complexity of the search becomes proportional to N times the number of terms in the sum over the columns. The number of terms is generally (depending on which LMBC code is employed) much less than the size of $\mathbf{g}$, so the search complexity can often be assumed to be of the same order of magnitude as N. (Note that N is the number of terms that have to be searched in ordinary memoryless VQ encoding of a size-N VQ).

5. SPEECH QUANTIZATION

To examine the performance of the described method we will in this section present experiments on quantization of speech samples. We have chosen to work with speech samples for simplicity and in order to illustrate the performance of the proposed method. Parameters extracted from speech, as well as blocked speech samples, generally have high correlation between consecutive vectors. Quantization schemes exploiting this property by linear predictive methods have indeed shown improved performance compared to memoryless methods. Occasionally, however, consecutive vectors have very low mutual dependency, corresponding to rapid changes of the speech signal. In these cases it has been shown advantageous to incorporate an additional robustness to this rapid change in the vector trajectory [9]. In [2], it is indicated that non-linear decoding, of the same form as (4), also inherits a robustness towards the varying intervector dependency. As previously mentioned, the method described here can be seen as a non-linear predictive vector quantizer. We will therefore compare the new approach to the performance of a traditional linear predictive scheme. In a common implementation of such a scheme [7], a linear prediction, $\tilde{\mathbf{x}}_n$, is given by

$$\tilde{\mathbf{x}}_n = \sum_{k=1}^{K} \mathbf{A}_k \hat{\mathbf{x}}_{n-k} \tag{13}$$

where $\hat{\mathbf{x}}_{n-k}$ are previously coded vectors, $\mathbf{A}_k$ are prediction coefficient matrices and K is the predictor order. The prediction error $\mathbf{e}_n = \mathbf{x}_n - \tilde{\mathbf{x}}_n$ is then quantized using a memoryless VQ.

The nonlinear prediction presented in this work has a finite memory M in the decoder, and linear predictive schemes that utilize a finite memory have been presented in the literature, e.g. [10, 11]. Since the decoding structure of a linear predictive quantizer based on (13) is recursive, there is an infinite memory in the

[1] This statement holds for the E3 code, and all other codes (like the ones proposed in [5, 4]) that are defined according to: "Include in $\mathbf{g}(\mathbf{i})$ all linear terms, all terms of weight w_1 plus all terms of weight $w_2 \ldots$", but does not hold for all LMBC codes.

decoder. We can thus regard the performance of a recursive structure as an upper bound on the performance of a finite memory structure.

Two sets of speech material were prepared from the TIMIT speech database for the experiment. The training was performed on 150 seconds and another 50 seconds was used for evaluation. For the linear predictive VQs a first order prediction is employed. The results of the experiments are shown in Table 1 for rate $R = L/d = 1$ and in Table 2 for $R = 2$. The tables show the obtained SNR (in dB) for memoryless quantization, linear prediction, optimal encoding-decoding (according to (7) and (4), indicated as "Full"(-freedom) in the table) and LMBC based encoding-decoding (according to (12) and (11) and employing the E3 code).

VQ	SNR [dB]
Mem.less	6.6
Lin. pred.	7.4
Full M=1	7.7
E3 M=1	7.6
Full M=2	8.1
E3 M=2	8.0
E3 M=3	8.3

Table 1: Performance at 5 bits and 5 dim.

VQ	SNR [dB]
Mem.less	10.7
Lin. pred.	11.7
Full M=1	12.2
E3 M=1	12.1
E3 M=2	13.0

Table 2: Performance at 6 bits and 3 dim.

As can be observed all memory schemes outperform memoryless quantization. Furthermore, the new approach performs better than linear predictive coding. We also note that the LMBC based method performs relatively close to the optimal approach.

In the experiments, we have noted that the LMBC-based approach is much more robust against discrepancies between the training set and the evaluation set, than is the optimal approach. This is most likely due to the fact that in the LMBC-based method the parameter to be estimated from the training set is the mapping matrix $\mathbf{G}$, while design of an optimal code is equivalent to estimating the full-size matrix $\mathbf{T}$ (or, equivalently, the set $\{\mathbf{v}(\mathbf{i})\}$ of decoder vectors). Since the size of the constrained matrix $\mathbf{G}$ is (much) less than the size of $\mathbf{T}$, in effect the unknown parameters to be estimated are much fewer in the LMBC approach. Hence the accuracy of the training, based on a limited set of training vectors, can be expected to be (much) higher in the LMBC-based approach, while a drawback of the optimal approach is that a very large training set is needed in order to get reliable results.

6. CONCLUSIONS AND FURTHER WORK

In this paper we have proposed a vector quantization method utilizing the dependency between transmitted indices. We have presented both an MMSE optimal decoder, given a collection of previously transmitted indices, and an MMSE optimal encoder for such a decoder. An optimal decoder needs a storage of possible reconstruction vector which grows exponentially with the size of the employed memory. We have therefore proposed a sub-optimal decoding scheme where the storage requirement at the decoder is substantially reduced. This is achieved by using a linear mapping of a block code to describe an approximation to the set of possible decoder vectors. By using a Hadamard formulation the proposed method can be interpreted as a nonlinear predictive vector quantization scheme of finite memory. Experiments on quantized speech showed that both the proposed optimal and sub-optimal schemes outperform a traditional linear predictive scheme. These promising results indicate that the proposed method will also be well suited for quantization of parameters extracted from speech, such as spectrum and excitation parameters. One main suggestion for further work is hence the application of the proposed method in parametric speech coding.

7. REFERENCES

[1] M. Skoglund, "Soft decoding for vector quantization over noisy channels with memory," *Subm. to IEEE Transactions on Information Theory*, March 1997.

[2] W. B. Kleijn, "On optimal and minimum-entropy decoding," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Processing*, Munich, 1997, pp. 1671–1674.

[3] A. Gersho, "Optimal nonlinear interpolative vector quantization," *IEEE Trans. on Communications*, vol. 38, no. 9-10, pp. 1285–1287, 1990.

[4] R. Hagen, "Robust LPC spectrum quantization - vector quantization by a linear mapping of a block code," *IEEE Transactions on Speech and Audio Processing*, vol. 4, no. 4, pp. 266–280, 1996.

[5] R. Hagen and P. Hedelin, "Robust vector quantization by a linear mapping of a block code," *Subm. to IEEE Transactions on Information Theory*, December 1995.

[6] P. Hedelin, P. Knagenhjelm, and M. Skoglund, "Theory for transmission of vector quantization data," in *Speech coding and synthesis*, K. K. Paliwal W. B. Kleijn, Ed. Elsevier Science, 1995.

[7] A. Gersho and R. M. Gray, *Vector quantization and signal compression*, Kluwer Academic Publishers, 1992.

[8] W-Y. Chan J. H. Y. Loo and P Kabal, "Classified nonlinear predictive vector quantization of speech spectral parameters," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Processing*, Atlanta, 1996, pp. 761–754.

[9] T. Eriksson, J. Lindén, and J. Skoglund, "Exploiting interframe correlation in spectral quantization - a study of different memory VQ schemes," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Processing*, Atlanta, 1996, pp. 765–768.

[10] H. Ohmuro, T. Moriya, K. Mano, and S. Miki, "Vector quantization of LSP parameters using moving average interframe prediction,," *Electronics and Communications in Japan, Part 3*, vol. 77, pp. 12–26, 1994.

[11] J. Skoglund and J. Lindén, "Predictive VQ for noisy channel spectrum coding: AR or MA?," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Processing*, Munich, 1997, pp. 1351–1354.