Linköping University Post Print

On Multiple Description Coding of Sources with Memory

Daniel Persson and Thomas Eriksson

N.B.: When citing this work, cite the original article.

©2010 IEEE. Personal use of this material is permitted. However, permission to reprint/republish this material for advertising or promotional purposes or for creating new collective works for resale or redistribution to servers or lists, or to reuse any copyrighted component of this work in other works must be obtained from the IEEE.

Daniel Persson and Thomas Eriksson, On Multiple Description Coding of Sources with Memory, 2010, IEEE Transactions on Communications, (58), 8, 2242-2251.

<http://dx.doi.org/10.1109/TCOMM.2010.070610.080689>

Postprint available at: Linköping University Electronic Press

<http://urn.kb.se/resolve?urn=urn:nbn:se:liu:diva-53733>

On Multiple Description Coding of Sources with Memory

Daniel Persson and Thomas Eriksson

Abstract—We propose a framework for multiple description coding (MDC) of sources with memory. A new source coding method for lossless transmission of correlated sources, power series quantization (PSQ), was recently suggested. PSQ uses a separate linear or non-linear predictor for each quantizer region, and has shown increased performance compared to several common quantization schemes for sources with memory. We propose multiple description PSQ as a special case within our framework. The suggested scheme is shown to increase performance compared with previous state-of-the-art MDC methods.

Index Terms—Channel-optimized quantization, packet erasure channel, memory-based quantization.

I. INTRODUCTION

IN THIS paper we consider transmission of a source with memory, linear or non-linear, over a lossy channel, which is relevant for e.g., speech, audio, and video transmission.

Efficient encoding of sources with memory may be provided by vector quantization (VQ) [1], but the exponential increase with dimension of codebook size, and thus also of storage and search complexity, renders VQ inconvenient for some applications. Another way of exploiting signal memory upon transmission is *memory-based quantizers*, that incorporate knowledge of previously quantized and transmitted data in the encoding process [2]. A memory-based VQ system always performs better than traditional VQ with the same vector dimension and codebook size, since attention is given also to inter-vector correlation.

Previous memory-based quantization alternatives are e.g., finite-state vector quantization (FSVQ) [3], differential pulse code modulation (DPCM) [2], predictive vector quantization (PVQ) [4], [5], vector predictive quantization [6], [7], safety-net quantization [8], and Gaussian mixture modeling (GMM) [9].

For transmission over a channel without losses, a new source coding method, power series quantization (PSQ) [10], that utilizes codebooks of arbitrary functions of previously quantized data, expressed as power series expansions, outperformed the VQ, FSVQ, PVQ, and safety-net PVQ methods in

terms of compression, with only a small increase in memory requirement and computational complexity. This has inspired us to try to extend the same method to also work under degraded channel conditions. Contrary to DPCM and PVQ, PSQ views prediction and quantization as one single problem.

All memory-based quantizers experience difficulties during transmission over lossy channels. Since these methods use previously quantized data for encoding, and since the encoder in general has no knowledge of which errors occur on the channel, the encoder and decoder will be unsynchronized, and errors will propagate.

Over the past years, several techniques for combatting packet losses have been proposed, e.g., automatic repeat-request (ARQ), layered coding, error concealment (EC), forward error correction (FEC), and multiple description coding (MDC) [11]–[13]. Though adequate for many applications, ARQ may cause time delays, and layered coding requires the base layer to be received without errors. FEC can only correct a certain number of bits, but performance drops rapidly thereafter. With FEC, the source and channel coding efforts are separate. For infinite VQ dimension and infinite channel block length, it is optimal to design source and channel coders individually [14]. However, for reasons of computational complexity and delay requirements, only finite VQ dimensions and block lengths are considered in practice, and in this setting, joint source-channel approaches increase performance compared to separate efforts.

MDC is a joint source-channel coding method where each description provides acceptable quality separately, and all descriptions together render the source well. Error concealment may be seen as a special action at the memory-based MDC decoder, namely the decoder treatment when all descriptions are lost. MDC has a well-established history in source coding. In [15], an achievable MDC rate-distortion region for general memoryless sources was found. This region was proved to be tight for the memoryless Gaussian source and squared error criterion in [16]. It has however been proven [17] that the achievable region in [15] is not tight in general. Inner and outer bounds of the rate-distortion region were established in [18] for a general stationary source with memory and mean squared error distortion measure. Further it is shown that the bounds become tight at high resolution.

An MDC scalar quantizer was proposed in [19], entropy-constrained scalar quantization was assessed in [20], and multiple description vector quantization (MDVQ) was presented in [21]. Transform MDC was investigated in [22]–[25], multiple description lattice quantizers were described in, e.g., [26]–[28], and filter banks for MDC was explored in [29].

Paper approved by Z. Xiong, the Editor for Distributed Coding and Processing of the IEEE Communications Society. Manuscript received February 5, 2009; revised April 22, 2009 and November 20, 2009.

D. Persson is with the Department of Electrical Engineering, Linköping University, S-581 83 Linköping, Sweden (e-mail: daniel.p@isy.liu.se).

T. Eriksson is with the Department of Signals and Systems, Chalmers University of Technology, S-412 96 Göteborg, Sweden (e-mail: thomase@chalmers.se).

This work was supported by the Swedish Foundation for Strategic Research (SSF).

Digital Object Identifier 10.1109/TCOMM.2010.070610.080689

Memory-based MDC methods have also been considered. Trellis-coded multiple description quantization was investigated in [30], [31], and [32]. Differential pulse code modulation systems based on channel-splitting were devised in [33] and [34]. In [35], multiple description linear predictive vector quantization (MDPVQ) was suggested, and it was shown that this scheme increases performance compared to the method in [34]. Linear PVQ outperformed FSVQ in terms of compression, number of floating point operations, and memory requirement in [10]. A further motivation for the method in [35] is the success of linear predictor-based real-world schemes, e.g., adaptive DPCM for speech coding [36], and inter-frame prediction in video coding [37]. The linear predictor of [35] is however a constraint, and the scheme cannot handle non-linear effects.

A. Goal and outline

A framework for MDC of sources with memory is established. Error concealment falls out as a special case of the treatment. Within the framework a multiple description power series quantization (MDPSQ) strategy is proposed. A low-complexity optimal MDPSQ encoder index search algorithm, as well as an off-line codebook optimization procedure, are derived. The proposed system generalizes [35] to sources with non-linear memory. We evaluate MDPSQ for a Gauss-Markov source, a non-linear synthetic source, as well as for a real-world application, namely line spectral frequencies (LSF) coding [38].

In previous research [39], we have derived an optimal PSQ-based coder for noisy channels without memory, that outperformed both channel-optimized VQ and PVQ. Different from our previous paper [39], this paper deals with packet erasure channels, using the multiple description coder. Moreover, the transmission of several source vectors in the same packet in this paper is modeled in terms of channels with memory, which is an extension of the treatment of memoryless channels in [39].

The remainder of this paper is organized as follows. In Section II, the problem we want to solve is specified. Section III describes our MDC framework for sources with memory. MDPSQ is addressed in Section IV. The system is simulated in Section V, and the paper is finally concluded in Section VI.

II. PROBLEM SPECIFICATION

For clarity of the presentation of the material in this paper, we begin by stating which problem we want to solve. The listed conditions describe a relevant standard setting.

- A. The source produces discrete time, continuous amplitude, and zero mean vectors, with linear or nonlinear inter-vector correlation.
- B. The channel and source are independent.
- C. The packet erasures are independent.
- D. The encoder does not access any information about the channel.
- E. The decoder has full information about the channel realizations.
- F. One source vector is encoded and decoded at a time, and how this encoding and decoding influence future quantization is not considered.

Fig. 1. Overview of the MDC system.

G. Two equal rate MDC descriptions are used.

H. The squared Euclidean distance measure is used for assessing performance.

The listed problem specifications will be expressed in mathematical terms and employed in the development in the next section. Whenever a problem specification is used, this will be clearly stated by a reference to the listed point in question.

III. MDC FOR SOURCES WITH MEMORY

The source described in (A) in Section II produces discrete time, continuous amplitude, and zero mean vectors $\mathbf{x}_n$, $n = 1, \dots, N$, with linear or non-linear correlation, see Fig. 1. We regard a multiple description strategy, where the signal vector $\mathbf{x}_n$ is quantized at time n , see (F) in Section II, by choosing a central codebook index $i_{0,n} \in \{1, \dots, M_0\}$, $M_0 \leq M$, a side codebook 1 index $i_{1,n} \in \{1, \dots, \sqrt{M}\}$, and a side codebook 2 index $i_{2,n} \in \{1, \dots, \sqrt{M}\}$. Using less than all possible codebook entries M for the central index is a means for introducing redundancy on a lossy channel. As stated in (G) in Section II, we consider equal rate side descriptions, which means that the side codebooks contain the same number of code vectors $\sqrt{M}$. The central and side codebooks are related by maps $i_{1,n} = f_1(i_{0,n})$, $i_{2,n} = f_2(i_{0,n})$, and $i_{0,n} = f_0(i_{1,n}, i_{2,n})$, for $i_{1,n} = 1, \dots, \sqrt{M}$, $i_{2,n} = 1, \dots, \sqrt{M}$, and $i_{0,n} = 1, \dots, M_0$. Indices actually chosen by the encoder at time n are $i_{0,n}^*$, $i_{1,n}^*$, and $i_{2,n}^*$. We introduce a vector of all sent central indices until and including time n

$$\mathbf{i}_{0,n}^* = [i_{0,1}^*, \dots, i_{0,n}^*]. \quad (1)$$

The central index $i_{0,n}^*$ is communicated only through transmission of $i_{1,n}^*$ and $i_{2,n}^*$ to the decoder, see Fig. 1. Each packet only contains indices from one of the two descriptions, i.e., side indices $i_{1,n}^*$ and $i_{2,n}^*$ are always transported in different packets. The packetization is synchronized in time so that indices $i_{1,n-1}^*$ and $i_{1,n}^*$ are transmitted in the same packet if and only if indices $i_{2,n-1}^*$ and $i_{2,n}^*$ are transmitted in the same packet. Moreover, the indexes are packetized in sequence, so that $i_{0,n'}^*$ is packetized before $i_{0,n}^*$ if $n' < n$. Packets containing description 1 indices and packets containing description 2 indices may be sent over the same physical channel, which is referred to as time diversity, or over two separate channels, where each channel only conveys packets pertaining to one description, which is referred to as path diversity.

As seen in Fig. 1, four possible *loss situations* l_n may occur at the decoder for the central index $i_{0,n}^*$ at time n . The decoder

has full information of the loss situation, as stated in (E), *i.e.*, l_n , as well as a quantizer index, are always delivered at the decoder:

Central quantizer, $l_n = 0$: Both $i_{1,n}^*$ and $i_{2,n}^*$ are delivered.

Side quantizer 1, $l_n = 1$: Only $i_{1,n}^*$ is delivered.

Side quantizer 2, $l_n = 2$: Only $i_{2,n}^*$ is delivered.

Error concealment, $l_n = 3$: Neither $i_{1,n}^*$ nor $i_{2,n}^*$ is delivered. We introduce an index $i_{3,n} = 1$ for the error concealment case for consistency with the other loss situations. This notation will simplify the equations greatly in later developments. Observe that there is only one possible index in the case of error concealment, and $i_{3,n}^* = 1$ for all n .

In the following, we will for simplicity refer to loss situation l_n instead of describing what happens to the two packets that communicate $i_{1,n}^*$ and $i_{2,n}^*$ at time n . The vector of all loss situations until and including time n is

$$\mathbf{l}_n = [l_1, \dots, l_n] \quad (2)$$

and the vector of received indices until and including time n is

$$\mathbf{j}_n = [i_{l_1,1}^*, \dots, i_{l_n,n}^*]. \quad (3)$$

The decoder handles one source vector at a time, ignores future quantization, has knowledge of the arrival status of each packet, see (F) and (E) in Section II, and the reconstruction is

$$\tilde{\mathbf{x}}_n = \tilde{\mathbf{x}}_n(\mathbf{l}_n, \mathbf{j}_n). \quad (4)$$

Moreover, as was specified in (D) in Section II, at time n , the encoder has knowledge about the source vector $\mathbf{x}_n$, and chosen indices $i_{0,n-1}^*$, but not the channel realizations $\mathbf{l}_n$. We may thus write the distortion measure as

$$\bar{D}_n(i_{0,n}) = E_{\mathbf{l}_n | \mathbf{x}_n, i_{0,n-1}^*} [D(\mathbf{x}_n, \tilde{\mathbf{x}}_n)] \quad (5)$$

$$= E_{\mathbf{l}_n | \mathbf{x}_n, i_{0,n-1}^*} [\|\mathbf{x}_n - \tilde{\mathbf{x}}_n\|_2^2], \quad (6)$$

where $D(\mathbf{x}_n, \tilde{\mathbf{x}}_n)$ is the distance measure, chosen in (6) to be the squared Euclidean distance according to (H) in Section II. The optimal encoder can thus be written as

$$i_{0,n}^* = \arg \min_{i_{0,n} \in \{1, \dots, M_0\}} \bar{D}_n(i_{0,n}) \quad (7)$$

$$= \arg \min_{i_{0,n} \in \{1, \dots, M_0\}} E_{\mathbf{l}_n | \mathbf{x}_n, i_{0,n-1}^*} [\|\mathbf{x}_n - \tilde{\mathbf{x}}_n\|_2^2]. \quad (8)$$

Observe that while choosing $i_{0,n}^*$, the side indexes are also defined by the maps $i_{1,n}^* = f(i_{0,n}^*)$ and $i_{2,n}^* = f(i_{0,n}^*)$. According to (B) and (D) in Section II, the source and encoder are independent of the channel

$$P(\mathbf{l}_n | \mathbf{x}_n, i_{0,n-1}^*) = P(\mathbf{l}_n) \quad (9)$$

and we may rewrite (8) as

$$i_{0,n}^* = \arg \min_{i_{0,n} \in \{1, \dots, M_0\}} E_{\mathbf{l}_n} [\|\mathbf{x}_n - \tilde{\mathbf{x}}_n\|_2^2]. \quad (10)$$

We consider independent packet losses, see (C). Note that though this channel is independent on the inter-packet level, there is memory on the intra-packet level, *i.e.*, a packet may carry several consecutive side indices, whose loss statuses are then correlated. When the packets carrying the side indices

Fig. 2. MDPSQ coder.

$i_{1,n}^*$ and $i_{2,n}^*$ at time n are the same as the packets carrying the side indices $i_{1,n-1}^*$ and $i_{2,n-1}^*$ at time $n-1$, l_n and l_{n-1} are identical, and when the packets carrying the side indices $i_{1,n}^*$ and $i_{2,n}^*$ at time n are different from the packets carrying the side indices $i_{1,n-1}^*$ and $i_{2,n-1}^*$ at time $n-1$, l_n and l_{n-1} are independent.

IV. MULTIPLE DESCRIPTION POWER SERIES QUANTIZATION (MDPSQ)

In this section, a new method, MDPSQ, is proposed, using the framework in Section III. Before MDPSQ is described, we review VQ, PVQ, and PSQ for a lossless channel [10], within the framework of Section III, as an introduction.

A. VQ for a lossless channel

When the channel is lossless, the index $i_{n,0}^*$ arrives correctly to the decoder for all n . In the case of VQ, the decoder uses a codebook $\mathbf{c}_m$, $m = 1, \dots, M$, that is fixed in time, and the decoder reconstruction of $\mathbf{x}_n$ at time n is

$$\tilde{\mathbf{x}}_n(i_{0,n}^*) = \mathbf{c}_{i_{0,n}^*}. \quad (11)$$

Using the codebook and (11), the encoder in (10) can be rewritten as

$$i_{0,n}^* = \arg \min_{m \in \{1, \dots, M\}} \|\mathbf{x}_n - \mathbf{c}_m\|_2^2. \quad (12)$$

B. PVQ for a lossless channel

In traditional PVQ, a prediction $\mathbf{h}(\tilde{\mathbf{y}}_n)$, where $\tilde{\mathbf{y}}_n$ is a vector containing previously quantized samples, is first subtracted from the vector to be transmitted, and the residual is thereafter quantized by standard memoryless VQ using a codebook $\mathbf{u}_m$, $m = 1, \dots, M$. This is equivalent to usage of a decoder codebook

$$\mathbf{c}_m^{(n)} = \mathbf{u}_m + \mathbf{h}(\tilde{\mathbf{y}}_n), \quad m = 1, \dots, M. \quad (13)$$

Since the term $\mathbf{h}(\tilde{\mathbf{y}}_n)$ contains previously quantized information, PVQ is classified as a memory-based method. Including the term $\mathbf{h}(\tilde{\mathbf{y}}_n)$ in (13) also makes the PVQ codebook time-dependent, which is a major difference from the VQ codebook in Section IV-A. We also observe that the set of all possible VQ codebooks is a subset of the set of all codebooks (13). The PVQ decoder reconstruction of $\mathbf{x}_n$ at time n (4) is

$$\tilde{\mathbf{x}}_n(i_{0,n}^*) = \mathbf{u}_{i_{0,n}^*} + \mathbf{h}(\tilde{\mathbf{y}}_n). \quad (14)$$

Using the codebook (13) and the decoder (14), the encoder in (10) can be rewritten as

$$i_{0,n}^* = \arg \min_{m \in \{1, \dots, M\}} \|\mathbf{x}_n - (\mathbf{u}_{i_{0,n}^*} + \mathbf{h}(\tilde{\mathbf{y}}_n))\|_2^2. \quad (15)$$

C. Power series quantization (PSQ) for a lossless channel

The main difference between PVQ and PSQ is that whereas PVQ uses the same predictor $\mathbf{h}(\tilde{\mathbf{y}}_n)$ for all quantization regions, see (13) and (14), PSQ uses a separate predictor, linear or nonlinear, in each region of the quantizer. The PSQ predictor functions are expressed as power series expansions. Since a power series expansion can describe almost all reasonably smooth functions, it can be shown that the PSQ can describe any type of nonlinear or linear memory. We write the PSQ decoder codebook

$$\mathbf{c}_m^{(n)} = \mathbf{A}_m \tilde{\mathbf{y}}_n, \quad m = 1, \dots, M, \quad (16)$$

where the matrices $\mathbf{A}_m$ contain the power series expansion coefficients, and $\tilde{\mathbf{y}}_n$ contains previously quantized scalar samples, powers thereof, as well as multiplications of such powers. Each column of $\mathbf{A}_m$ corresponds to a power series expansion coefficient. The number of columns in $\mathbf{A}_m$ is determined by the power series expansion order, the number of previously quantized source vectors used for quantization at the present time, as well as the dimension of these vectors. At the decoder side, the reconstruction of $\mathbf{x}_n$ at time n (4) is

$$\tilde{\mathbf{x}}_n = \mathbf{A}_{i_{0,n}^*} \tilde{\mathbf{y}}_n. \quad (17)$$

Consider for example the decoding at time n using one vector memory

$$\tilde{\mathbf{x}}_{n-1} = [\tilde{x}_{n-1,1}, \tilde{x}_{n-1,2}]^T, \quad (18)$$

where $\tilde{x}_{n-1,1}$, and $\tilde{x}_{n-1,2}$ are scalar vector components. If we use a second order power series expansion, we can write

$$\tilde{\mathbf{y}}_n = [1, \tilde{x}_{n-1,1}, \tilde{x}_{n-1,2}, (\tilde{x}_{n-1,1})^2, (\tilde{x}_{n-1,2})^2, \tilde{x}_{n-1,1}\tilde{x}_{n-1,2}]^T. \quad (19)$$

Equations (17) and (19) define a recursion, and we may thus write $\tilde{\mathbf{y}}_n = \tilde{\mathbf{y}}_n(i_{0,n-1}^*) = \tilde{\mathbf{y}}_n(\tilde{\mathbf{x}}_1, \dots, \tilde{\mathbf{x}}_{n-1})$ and $\tilde{\mathbf{x}}_n = \tilde{\mathbf{x}}_n(i_{0,n}^*) = \tilde{\mathbf{x}}_n(\tilde{\mathbf{x}}_1, \dots, \tilde{\mathbf{x}}_{n-1}, i_{0,n}^*)$. A key observation is that (16) has more degrees of freedom than the PVQ codebook (13), for the same codebook size M . PSQ based on a first order power series expansion, with one vector memory

$$\tilde{\mathbf{y}}_n = \begin{bmatrix} 1 \\ \tilde{\mathbf{x}}_{n-1} \end{bmatrix} \quad (20)$$

is regarded in this paper for handling nonlinear correlation in the source sequence, since this previously has showed good

performance in applications [10]. Using (16) and (17), the encoder in (10) can be rewritten as

$$i_{0,n}^* = \arg \min_{m \in \{1, \dots, M\}} \|\mathbf{x}_n - \mathbf{A}_m \tilde{\mathbf{y}}_n\|_2^2. \quad (21)$$

For a more thorough introduction to PSQ, we refer the reader to [10].

D. MDPSQ coder

The PSQ scheme in Section (IV-C) is now expanded in order to tackle transmission of a source with memory over a channel with packet erasures, which is the general situation described in Section III. The decoder codebooks are

$$\mathbf{c}_m^{(n)} = \mathbf{A}_m^{(l_n)} \tilde{\mathbf{y}}_n, \quad (22)$$

where $m = 1, \dots, M_0$ if $l_n = 0$, $m = 1, \dots, \sqrt{M}$ if $l_n = 1, 2$, $m = 1$ if $l_n = 3$, and $\tilde{\mathbf{y}}_n = \tilde{\mathbf{y}}_n(l_{n-1}, j_{n-1})$ now is a function of the loss situations and the received indices, see Fig. 2. Our decoder reconstruction (4) is

$$\tilde{\mathbf{x}}_n = \mathbf{A}_{i_{l_n,n}^*}^{(l_n)} \tilde{\mathbf{y}}_n. \quad (23)$$

Encoding is most often performed online, and has to be reasonably fast. Evaluating all expectations in (10) for each n would result in a coder whose complexity increases in time. Our goal in this section is to formulate the encoder (10) as a low-complexity algorithm where calculations for encoding at time $n-1$ can be reused in a recursive manner for encoding at time n . We begin by making the necessary developments of (10), in order to be able to state the algorithm at the end of the section

$$i_{0,n}^* = \arg \min_{i_{0,n} \in \{1, \dots, M_0\}} \left(E_{l_n}[\mathbf{x}_n^T \mathbf{x}_n] - 2\mathbf{x}_n^T \underbrace{E_{l_n}[\tilde{\mathbf{x}}_n]}_{\mathbf{s}(i_{0,n})} + \underbrace{E_{l_n}[\tilde{\mathbf{x}}_n^T \tilde{\mathbf{x}}_n]}_{r(i_{0,n})} \right). \quad (24)$$

The first term in (24) is the same for all choices of $i_{0,n}^*$, and is neglected. We will now find efficient ways of calculating the expectations in $\mathbf{s}(i_{0,n})$ and $r(i_{0,n})$.

$\mathbf{s}(i_{0,n})$: We introduce

$$\mathbf{v}_n(l_n, j_n) = E_{l_{n-1}|l_n}[\tilde{\mathbf{x}}_n] \quad (25)$$

$$= E_{l_{n-1}|l_n}[\mathbf{A}_{i_{l_n,n}^{(l_n)}} \tilde{\mathbf{y}}_n] \quad (26)$$

$$= \mathbf{A}_{i_{l_n,n}^{(l_n)}} E_{l_{n-1}|l_n}[\tilde{\mathbf{y}}_n] \quad (27)$$

$$= \mathbf{A}_{i_{l_n,n}^{(l_n)}} E_{l_{n-1}|l_n} \left[\begin{bmatrix} 1 \\ \tilde{\mathbf{x}}_{n-1} \end{bmatrix} \right] \quad (28)$$

$$= \mathbf{A}_{i_{l_n,n}^{(l_n)}} E_{l_{n-1}|l_n} \left[\begin{bmatrix} 1 \\ E_{l_{n-2}|l_{n-1}, l_n}[\tilde{\mathbf{x}}_{n-1}] \end{bmatrix} \right] \quad (29)$$

$$= \mathbf{A}_{i_{l_n,n}^{(l_n)}} E_{l_{n-1}|l_n} \left[\begin{bmatrix} 1 \\ E_{l_{n-2}|l_{n-1}}[\tilde{\mathbf{x}}_{n-1}] \end{bmatrix} \right] \quad (30)$$

$$= \mathbf{A}_{i_{l_n,n}^{(l_n)}} E_{l_{n-1}|l_n} \left[\begin{bmatrix} 1 \\ \mathbf{v}_{n-1}(l_{n-1}, j_{n-1}) \end{bmatrix} \right], \quad (31)$$

where we have used (22) to obtain (26). The fact that matrix multiplication is a linear map, and that l_n is given, results in (27), (20) is used to obtain (28), and standard manipulation of the probabilities results in (29). If $i_{0,n-1}^*$ and $i_{0,n}^*$ are carried by the same packet pair, i.e., if l_{n-1} and l_n are identical, we can write (30). Otherwise $i_{0,n}^*$ and $i_{0,n'}^*$ for $n' < n$ are carried by different packet pairs, which means that l_{n-2} and l_{n-1} are independent of l_n , in which case (30) also holds. Equation (25) is employed for achieving (31). Thus, we have derived a recursive way to calculate $\mathbf{v}_n$ from $\mathbf{v}_{n-1}$. We evaluate (31) by treating the cases where l_n and l_{n-1} are identical and independent separately:

- In the case where l_n and l_{n-1} are identical we can rewrite (31) as

$$\mathbf{v}_n(l_n, j_n) = \mathbf{A}_{i_{l_n,n}}^{(l_n)} \begin{bmatrix} 1 \\ \mathbf{v}_{n-1}(l_{n-1}, j_{n-1}) \end{bmatrix}. \quad (32)$$

- In the case where l_n and l_{n-1} are independent we can rewrite (31) as

$$\mathbf{v}_n(l_n, j_n) = \mathbf{A}_{i_{l_n,n}}^{(l_n)} \cdot E_{l_{n-1}} \left[\begin{bmatrix} 1 \\ \mathbf{v}_{n-1}(l_{n-1}, j_{n-1}) \end{bmatrix} \right]. \quad (33)$$

The term $s(i_{0,n})$ in (24) can finally be rewritten as

$$s(i_{0,n}) = E_{l_n}[\mathbf{v}_n(l_n, j_n)]. \quad (34)$$

$r(i_{0,n})$: We introduce

$$\mathbf{W}_n(l_n, j_n) = E_{l_{n-1}|l_n}[\tilde{\mathbf{x}}_n \tilde{\mathbf{x}}_n^T] \quad (35)$$

$$= E_{l_{n-1}|l_n} \left[\mathbf{A}_{i_{l_n,n}}^{(l_n)} \tilde{\mathbf{y}}_n \tilde{\mathbf{y}}_n^T (\mathbf{A}_{i_{l_n,n}}^{(l_n)})^T \right] \quad (36)$$

$$= \mathbf{A}_{i_{l_n,n}}^{(l_n)} E_{l_{n-1}|l_n}[\tilde{\mathbf{y}}_n \tilde{\mathbf{y}}_n^T] (\mathbf{A}_{i_{l_n,n}}^{(l_n)})^T \quad (37)$$

$$= \mathbf{A}_{i_{l_n,n}}^{(l_n)} E_{l_{n-1}|l_n} \left[\begin{bmatrix} 1 & \tilde{\mathbf{x}}_{n-1}^T \\ \tilde{\mathbf{x}}_{n-1} & \tilde{\mathbf{x}}_{n-1} \tilde{\mathbf{x}}_{n-1}^T \end{bmatrix} \right] (\mathbf{A}_{i_{l_n,n}}^{(l_n)})^T \quad (38)$$

$$= \mathbf{A}_{i_{l_n,n}}^{(l_n)} E_{l_{n-1}|l_n} \left[\begin{bmatrix} 1 & (E_{l_{n-2}|l_{n-1}, l_n}[\tilde{\mathbf{x}}_{n-1}])^T \\ E_{l_{n-2}|l_{n-1}, l_n}[\tilde{\mathbf{x}}_{n-1}] & E_{l_{n-2}|l_{n-1}, l_n}[\tilde{\mathbf{x}}_{n-1} \tilde{\mathbf{x}}_{n-1}^T] \end{bmatrix} \right] (\mathbf{A}_{i_{l_n,n}}^{(l_n)})^T \quad (39)$$

$$= \mathbf{A}_{i_{l_n,n}}^{(l_n)} E_{l_{n-1}|l_n} \left[\begin{bmatrix} 1 & (E_{l_{n-2}|l_{n-1}}[\tilde{\mathbf{x}}_{n-1}])^T \\ E_{l_{n-2}|l_{n-1}}[\tilde{\mathbf{x}}_{n-1}] & E_{l_{n-2}|l_{n-1}}[\tilde{\mathbf{x}}_{n-1} \tilde{\mathbf{x}}_{n-1}^T] \end{bmatrix} \right] (\mathbf{A}_{i_{l_n,n}}^{(l_n)})^T \quad (40)$$

$$= \mathbf{A}_{i_{l_n,n}}^{(l_n)} E_{l_{n-1}|l_n} \left[\begin{bmatrix} 1 & (E_{l_{n-2}|l_{n-1}}[\tilde{\mathbf{x}}_{n-1}])^T \\ E_{l_{n-2}|l_{n-1}}[\tilde{\mathbf{x}}_{n-1}] & E_{l_{n-2}|l_{n-1}}[\tilde{\mathbf{x}}_{n-1} \tilde{\mathbf{x}}_{n-1}^T] \end{bmatrix} \right] (\mathbf{A}_{i_{l_n,n}}^{(l_n)})^T$$

$$\left[\begin{bmatrix} 1 & \mathbf{v}_{n-1}^T(l_{n-1}, j_{n-1}) \\ \mathbf{v}_{n-1}(l_{n-1}, j_{n-1}) & \mathbf{W}_{n-1}(l_{n-1}, j_{n-1}) \end{bmatrix} \right] (\mathbf{A}_{i_{l_n,n}}^{(l_n)})^T, \quad (41)$$

where we have used (22) to obtain (36). The fact that matrix multiplication is a linear map, and that l_n is given, results in (37), (20) is used to obtain (38), and standard manipulations of the probabilities results in (39). If $i_{0,n-1}^*$ and $i_{0,n}^*$ are carried by the same packet pair, i.e., if l_{n-1} and l_n are identical, we can write (40). Otherwise $i_{0,n}^*$ and $i_{0,n'}^*$ for $n' < n$ are carried by different packet pairs, which means that l_{n-2} and l_{n-1} are independent of l_n , in which case (40) also holds. Thus, we have derived a recursive way to calculate $\mathbf{W}_n$ from $\mathbf{W}_{n-1}$. We now evaluate (41) by treating the cases where l_n and l_{n-1} are identical and independent separately:

- In the case where l_n and l_{n-1} are identical we can rewrite (41) as

$$\mathbf{W}_n(l_n, j_n) = \mathbf{A}_{i_{l_n,n}}^{(l_n)} \cdot \begin{bmatrix} 1 & \mathbf{v}_{n-1}^T(l_{n-1}, j_{n-1}) \\ \mathbf{v}_{n-1}(l_{n-1}, j_{n-1}) & \mathbf{W}_{n-1}(l_{n-1}, j_{n-1}) \end{bmatrix} \cdot (\mathbf{A}_{i_{l_n,n}}^{(l_n)})^T. \quad (42)$$

- In the case where l_n and l_{n-1} are independent we can rewrite (41) as

$$\mathbf{W}_n(l_n, j_n) = \mathbf{A}_{i_{l_n,n}}^{(l_n)} E_{l_{n-1}} \left[\begin{bmatrix} 1 & \mathbf{v}_{n-1}^T(l_{n-1}, j_{n-1}) \\ \mathbf{v}_{n-1}(l_{n-1}, j_{n-1}) & \mathbf{W}_{n-1}(l_{n-1}, j_{n-1}) \end{bmatrix} \right] (\mathbf{A}_{i_{l_n,n}}^{(l_n)})^T. \quad (43)$$

The term $r(i_{0,n})$ in (24) can finally be rewritten as

$$r(i_{0,n}) = E_{l_n}[\tilde{\mathbf{x}}_n^T \tilde{\mathbf{x}}_n] \quad (44)$$

$$= E_{l_n}[\text{tr}[\tilde{\mathbf{x}}_n \tilde{\mathbf{x}}_n^T]] \quad (45)$$

$$= \text{tr}[E_{l_n}[\tilde{\mathbf{x}}_n \tilde{\mathbf{x}}_n^T]] \quad (46)$$

$$= \text{tr}[E_{l_n}[E_{l_{n-1}|l_n}[\tilde{\mathbf{x}}_n \tilde{\mathbf{x}}_n^T]]] \quad (47)$$

$$= \text{tr}[E_{l_n}[\mathbf{W}_n(l_n, j_n)]], \quad (48)$$

where we have used that trace, denoted by tr , is a linear map in order to obtain (46), and (35) is used to obtain (48). We finally summarize our low-complexity encoder in Algorithm 1. The MDPSQ encoder is more computationally demanding than the MDPVQ encoder since it uses one predictor per quantization region. MDPSQ complexity can however be significantly reduced by only considering certain non-zero entries in the power series expansion matrices. By limiting the power series coefficients of order 1 to zero, MDVQ is achieved as a special case of MDPSQ. If the codebook power series matrices are such that coefficients of order 1 at the same matrix position are equal, MDPVQ is achieved. Strategies with less than M_0 different predictors, as well as safety-net MDPVQ, can be implemented by similar methodologies. Also, it should be noted that the MDPSQ encoder is highly parallelizable.

Algorithm 1 Encoder procedure.

- 1: Initialization: Set $n = 0$, $\mathbf{x}_0$, and $\mathbf{v}_0$ to all zero vectors, and $\mathbf{W}_0$ to an all zero matrix.
- 2: Set $n = n + 1$. The vector $\mathbf{x}_n$ arrives from the source to the encoder.
- 3: Calculate $\mathbf{v}_n(l_n, j_n)$ and $\mathbf{W}_n(l_n, j_n)$ for $l_n = 0, \dots, 3$ and $i_{0,n} = 1, \dots, M_0$. Use (32) and (42) if l_n and l_{n-1} are identical or (33) and (43) if l_n and l_{n-1} are independent.
- 4: Calculate $\mathbf{s}(i_{0,n})$ and $\mathbf{r}(i_{0,n})$ for $i_{0,n} = 1, \dots, M_0$ using (34) and (48).
- 5: Decide $i_{0,n}^*$ using (24). Also store $\mathbf{v}_n(l_n, j_n)$ and $\mathbf{W}_n(l_n, j_n)$ for $i_{0,n}^*$ and $l_n = 0, \dots, 3$ for usage at time $n + 1$.
- 6: Stop if $n = N$, or otherwise go to step 2.

E. MDPSQ codebook optimization

It is crucial that the codebooks are well adapted to the source and channel. In this section, we propose a sample iterative algorithm to be run off-line in order to minimize the distortion in (24) with respect to the codebooks.

The power series expansion coefficient matrices are now considered to be functions of time n , and the matrix notation is accordingly changed slightly to $\mathbf{A}_m^{(k)}(n)$, where $k = 0, 1, \dots, 3$ indicates loss situation, m is codebook index ($m = 1, \dots, M_0$ if $k = 0$, $m = 1, \dots, \sqrt{M}$ if $k = 1, 2$, and $m = 1$ if $k = 3$) as before, and $n = 0, \dots, N$ is the time index. At each time instant n , $\mathbf{x}_n$ is first encoded to $i_{0,n}^*$ using $\mathbf{A}_m^{(k)}(n)$. Thereafter, for minimization of the distortion in (24), the new coefficient matrices at time $n + 1$ are given by

$$\mathbf{A}_m^{(k)}(n+1) = \mathbf{A}_m^{(k)}(n) - \mu(n) \nabla_{\mathbf{A}_m^{(k)}(n)} \bar{D}_n(i_{0,n}^*) \quad (49)$$

for all m and k , where $\mu(n)$ is a scalar step size. Observe that the gradient is only taken with respect to $\mathbf{A}_m^{(k)}(n)$ for the codebook update at time $n + 1$, and codebooks at times before n are ignored. We rewrite the descent direction in (49) as

$$\nabla_{\mathbf{A}_m^{(k)}(n)} \bar{D}_n(i_{0,n}^*) = \nabla_{\mathbf{A}_m^{(k)}(n)} E_{l_n} [\|\mathbf{x}_n - \tilde{\mathbf{x}}_n\|_2^2] \quad (50)$$

$$= \nabla_{\mathbf{A}_m^{(k)}(n)} \left(E_{l_n} [\mathbf{x}_n^T \mathbf{x}_n] - 2 \mathbf{x}_n^T \underbrace{E_{l_n} [\tilde{\mathbf{x}}_n]}_{\mathbf{s}(i_{0,n}^*)} + \underbrace{E_{l_n} [\tilde{\mathbf{x}}_n^T \tilde{\mathbf{x}}_n]}_{r(i_{0,n}^*)} \right) \quad (51)$$

$$= -\nabla_{\mathbf{A}_m^{(k)}(n)} (2 \mathbf{x}_n^T \mathbf{s}(i_{0,n}^*) - r(i_{0,n}^*)). \quad (52)$$

The development differs depending on whether l_n and l_{n-1} are identical or independent:

- In the case where l_n and l_{n-1} are identical we can use (32), (34), (42), and (48) to rewrite (52) as (53).
- In the case where l_n and l_{n-1} are independent we can use (33), (34), (43), and (48) to rewrite (52) as (54).

We summarize the sample iterative codebook optimization in Algorithm 2. This optimization is such that indices which are chosen more often are given more attention in the training. Our sample-iterative algorithm minimizes the distortion in (6) or equivalently in (24), which changes at each time n , on a vector-by-vector basis, and there is no distortion measure that is valid for all times n , for which we could assess convergence.

The computational complexity for codebook training is not

Algorithm 2 Sample iterative codebook optimization.

- 1: Initialization: Set $n = 0$, $\mathbf{x}_0$, and $\mathbf{v}_0$ to all zero vectors, and $\mathbf{W}_0$ to an all zero matrix.
- 2: Set $n = n + 1$. The vector $\mathbf{x}_n$ arrives from the source to the encoder.
- 3: An optimal index $i_{0,n}^*$ is chosen by the encoder using the power series expansion codebooks with the coefficient matrices $\mathbf{A}_m^{(k)}(n)$.
- 4: The coefficient matrices $\mathbf{A}_m^{(k)}(n+1)$ are calculated using (49). The descent direction in (49) is given by (53) in the case when l_n and l_{n-1} are identical, and by (54) in the case when l_n and l_{n-1} are independent.
- 5: Stop if $n = N$, or otherwise go to step 2.

very high compared to the encoder, since the same quantities that are used in the encoding are also used for codebook training. Codebook training is typically performed offline.

An MDC problem may be stated in two ways, either with Lagrange multipliers, or in terms of probabilities, for giving the priorities to side and central distortions. We may run our system for a certain loss probability, but use another probability in the encoder and codebook training, to favor either side or central distortion. This yields the Lagrange and probability approaches equivalent. It is not the first time that MDC is expressed in terms of probabilities, see [35] for example.

V. EXPERIMENTS

In this section, the simulation performances of MDPSQ and other MDC systems are compared.

A. Prerequisites

The simulation details are as follows:

- **Benchmarking:** MDPSQ is compared to MDPVQ [35], and MDVQ [21]. Outer bounds of the multiple description distortion region [18] are compared to the simulations. For the calculation of the differential entropy rate, we used a Gaussian assumption, and three consecutive vector realizations.
- **Performance measures:** Our method aims at minimizing mean square error (MSE), which is equivalent to maximizing signal-to-noise ratio (SNR), and these measures are therefore used for assessing performance. LSF quantization is also evaluated using spectral distortion (SD), which is a well established measure of LPC coding quality [40]. Spectral distortion is calculated in the full 0-4 kHz range.
- **Codebook optimization:** We choose to use a step size $\mu(n)$ that decreases linearly with n to 0. The start value for the step size is 0.1 for power series coefficients of order 0, and 0.05 for power series coefficients of order 1. As discussed in Section IV-E, the sample-iterative algorithm is not associated with any distortion measure for which convergence can be assessed. The algorithm however showed beneficial for codebook training in preliminary investigations.

$$\nabla_{\mathbf{A}_m^{(k)}(n)} \bar{D}_n(i_{0,n}^*) = \begin{cases} -2P_n(k) \left(\mathbf{x}_n \begin{bmatrix} 1 & \mathbf{v}_{n-1}^T(l_{n-1}, j_{n-1}) \end{bmatrix} \right. \\ \left. - \mathbf{A}_m^{(k)}(n) \begin{bmatrix} 1 & \mathbf{v}_{n-1}^T(l_{n-1}, j_{n-1}) \\ \mathbf{v}_{n-1}(l_{n-1}, j_{n-1}) & \mathbf{W}_{n-1}(l_{n-1}, j_{n-1}) \end{bmatrix} \right), & \text{if } m = i_{k,n}^*, k = 0, \dots, 3, \\ 0, & \text{otherwise.} \end{cases} \quad (53)$$

$$\nabla_{\mathbf{A}_m^{(k)}(n)} \bar{D}_n(i_{0,n}^*) = \begin{cases} -2P_n(k) \left(\mathbf{x}_n E_{l_{n-1}} \begin{bmatrix} 1 & \mathbf{v}_{n-1}^T(l_{n-1}, j_{n-1}) \end{bmatrix} \right. \\ \left. - \mathbf{A}_m^{(k)}(n) E_{l_{n-1}} \begin{bmatrix} 1 & \mathbf{v}_{n-1}^T(l_{n-1}, j_{n-1}) \\ \mathbf{v}_{n-1}(l_{n-1}, j_{n-1}) & \mathbf{W}_{n-1}(l_{n-1}, j_{n-1}) \end{bmatrix} \right), & \text{if } m = i_{k,n}^*, k = 0, \dots, 3, \\ 0, & \text{otherwise.} \end{cases} \quad (54)$$

- **Initialization and index assignment (IA):** Power series coefficients of order 0 are initialized by small random numbers, and coefficients of order 1 are initialized by a linear predictor estimated from the training database. Thereafter, the simulated annealing index assignment (IA) method [41], that has showed good performance in comparison with other schemes, is used. This IA is repeated for $M_0 = \sqrt{M}, 0.25M, 0.5M, 0.75M$, and M . The IA that results in the best performance with the training data is used for subsequent channel-optimized MDPSQ training. Also MDPVQ and MDVQ are initialized in this way. As was discussed in [41], channel-optimized codebook training may also by itself yield good index assignments.

B. Results

Three different processes are investigated:

- **Gauss-Markov process:** We first quantize a process with linear memory, namely the Gauss-Markov process

$$z_n = 0.9z_{n-1} + r_n, \quad (55)$$

where the r_n are i.i.d. zero-mean with variance 1 for all n . The scalar samples are divided into vectors $\mathbf{x}_n$ with dimension 2 prior to quantization. Each packet carries one description of 4 vectors, and 1 bit per sample and description is used for quantization. Two different databases containing $5 \cdot 10^5$ vectors each are used for the training and the evaluation.

The results are shown in Fig. 3 and Fig. 4. MDPSQ and MDPVQ [35] that exploit the memory in the process perform better than MDVQ [21]. Moreover, MDPVQ and MDPSQ perform equally well for small loss probabilities, while MDPSQ performs better than MDPVQ for big loss probabilities. MDPSQ should not increase performance compared to MDPVQ for a linear source and small loss probabilities, since the single MDPVQ predictor captures all the memory in the process. With a lossy channel, MDPSQ can however, unlike MDPVQ, use some codewords with very weak predictors, that stop error propagation.

Fig. 3. SNR versus packet error rate for MDPSQ, MDPVQ, and MDVQ, in the case of the Gauss-Markov process. Each packet carries one description of 4 2-dimensional vectors, and 1 bit per sample and description is used for quantization.

- **Non-linear synthetic process:** We investigate the highly non-linear process

$$\mathbf{x}_n = \begin{cases} \begin{bmatrix} 0.9 & 0 \\ 0 & 0.9 \end{bmatrix} \mathbf{x}_{n-1} + 0.3 \begin{bmatrix} r_n^1 \\ r_n^2 \end{bmatrix} & \text{with prob. } 2/5, \\ -\begin{bmatrix} 0.9 & 0 \\ 0 & 0.9 \end{bmatrix} \mathbf{x}_{n-1} + 0.3 \begin{bmatrix} r_n^1 \\ r_n^2 \end{bmatrix} & \text{with prob. } 2/5, \\ \begin{bmatrix} r_n^1 \\ r_n^2 \end{bmatrix} & \text{with prob. } 1/5, \end{cases} \quad (56)$$

where r_n^1, r_n^2 are i.i.d. zero-mean with variance 1 for all n . Each packet carries one description of 4 2-dimensional vectors, and 1 bit per sample and description is used for quantization. Two different databases containing $5 \cdot 10^5$ vectors each are used for the training and the evaluation. In Fig. 5 and Fig. 6 it is seen that MDPSQ outperforms both MDPVQ and MDVQ for all loss probabilities. It is

Fig. 4. Side MSE versus central MSE for MDPSQ, MDPVQ, MDVQ, and the rate distortion bound [18], in the case of the Gauss-Markov process. Each packet carries one description of 4 2-dimensional vectors, and 1 bit per sample and description is used for quantization. The different measurement points correspond to packet error rates 10^{-3} , 10^{-2} , 0.05, and 0.1.

Fig. 6. Side MSE versus central MSE for MDPSQ, MDPVQ, MDVQ, and the rate distortion bound [18], in the case of the non-linear synthetic process. Each packet carries one description of 4 2-dimensional vectors, and 1 bit per sample and description is used for quantization. The different measurement points correspond to packet error rates 10^{-3} , 10^{-2} , 0.05, and 0.1.

Fig. 5. SNR versus packet error rate for MDPSQ, MDPVQ, and MDVQ, in the case of the non-linear synthetic process. Each packet carries one description of 4 2-dimensional vectors, and 1 bit per sample and description is used for quantization.

Fig. 7. Spectral distortion versus packet error rate for MDPSQ, MDPVQ, and MDVQ, in the case of LSF quantization. Each packet carries one description of a single LSF-vector, and 1.2 bits per sample and description are used for quantization.

also observed that MDPVQ and MDVQ perform equally well. From studying (56) it is also easily seen that there is little use for a linear predictor when describing this process.

- LSF process:** The TIMIT database (lowpass-filtered and downsampled to 8 kHz) is used. A tenth-order linear predictive coding (LPC) analysis with the auto-correlation method is performed every 20 ms using a 25-ms Hamming window. A fixed 10-Hz bandwidth expansion is applied to each pole of the LPC coefficient vector, and the LPC vectors are transformed to the LSF representation. For training and evaluation, two separate sets containing $7 \cdot 10^5$ and $2.5 \cdot 10^5$ vectors respectively are employed. The vectors are split into three parts prior to quantization, with dimensions 3, 3, and 4 respectively.

Fig. 7 and Fig. 8 show the result when each packet

carries a single description for a single LSF vector, and 1.2 bits per sample and description, i.e., a total bit-rate of 24 bits per LSF vector, or 4 bits per sub-vector and description, are used for quantization. Similarly, Fig. 9 and Fig. 10 show the result when each packet carries a single description for each of 4 consecutive LSF-vectors. MDPSQ is consistently better than MDPVQ and MDVQ, regardless of the rate. MDPVQ always performs better than MDVQ, and in conclusion, the more carefully the memory is exploited, the better the performance becomes. Figure 11 illustrates the robustness of MDPSQ in the case when each packet carries one description of a single LSF-vector. Optimization for a specific error rate gives the best performance in practice for the same error rate. Good robustness is achieved except when training is performed with low error rate, which leads to big performance loss

Fig. 8. Side MSE versus central MSE for MDPSQ, MDPVQ, MDVQ, and the rate distortion bound [18], in the case of LSF quantization. Each packet carries one description of a single LSF-vector, and 1.2 bits per sample and description are used for quantization. The different measurement points correspond to packet error rates 10^{-3} , 10^{-2} , 0.05, and 0.1.

Fig. 10. Side MSE versus central MSE for MDPSQ, MDPVQ, MDVQ, and the rate distortion bound [18], in the case of LSF quantization. Each packet carries a single description for each of 4 consecutive LSF-vectors, and 1.2 bits per sample and description are used for quantization. The different measurement points correspond to packet error rates 10^{-3} , 10^{-2} , 0.05, and 0.1.

Fig. 9. Spectral distortion versus packet error rate for MDPSQ, MDPVQ, and MDVQ, in the case of LSF quantization. Each packet carries a single description for each of 4 consecutive LSF-vectors, and 1.2 bits per sample and description are used for quantization.

Fig. 11. SNR versus packet error rate for illustration of robustness, in the case of LSF quantization. Each packet carries one description of a single LSF-vector, and 1.2 bits per sample and description are used for quantization. The different curves correspond to MDPSQ optimized for channel error probabilities 10^{-4} , 10^{-2} , 0.05, and 0.1.

when evaluating for high error rates.

For quantization of 3-dimensional vectors with $M_0 = M = 2^8$, and 1 vector per packet, the computational complexity is $6.4 \cdot 10^4$ floating point operations per vector for MDPSQ, and $1.5 \cdot 10^3$ floating point operations per vector for MDPVQ and MDVQ. However, as has already been discussed in Section IV-D, by restraining the power series expansions, MDPSQ can operate closer to, or as, MDPVQ and MDVQ complexity-wise. We may also shrink the number of vectors in the MDPSQ side codebooks, and operate at the same distortion as MDPVQ. In this case, transmission bits are saved as well as computational complexity.

VI. CONCLUSION

The problem of transmission of a source with memory over a packet erasure channel is studied. A newly suggested

quantization method PSQ has outperformed several previous state-of-the-art algorithms for encoding of correlated sources in loss-free scenarios. We propose a multiple description PSQ (MDPSQ) algorithm, where PSQ is used for quantization in combination with an MDC strategy for combatting packet losses.

A recursive encoder, as well as an off-line sample iterative codebook optimization scheme, are derived. By experiments, it is seen that MDPSQ outperforms previously proposed state-of-the-art schemes, and MDPSQ robustness is confirmed.

REFERENCES

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

- [2] N. Jayant and P. Noll, *Digital Coding of Waveforms: Principles and Applications to Speech and Video*. Englewood Cliffs, NJ: Prentice-Hall, 1984.
- [3] J. Foster, R. Gray, and M. Dunham, "Finite-state vector quantization for waveform coding," *IEEE Trans. Inf. Theory*, vol. 31, no. 3, pp. 348-359, May 1985.
- [4] P.-C. Chang and R. Gray, "Gradient algorithms for designing predictive vector quantizers," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. 34, no. 4, pp. 679-690, Aug. 1986.
- [5] A. Gersho, "Optimal nonlinear interpolative vector quantization," *IEEE Trans. Commun.*, vol. 38, no. 9, pp. 1285-1287, Sep. 1990.
- [6] V. Cuperman and A. Gersho, "Vector predictive coding of speech at 16 kbits/s," *IEEE Trans. Commun.*, vol. 33, no. 7, pp. 685-696, July 1985.
- [7] Y. Shoham, "Vector predictive quantization of the spectral parameters for low rate speech coding," in *Proc. ICASSP*, Apr. 1987, pp. 2181-2184, vol. 12.
- [8] T. Eriksson, J. Linden, and J. Skoglund, "Interframe LSF quantization for noisy channels," *IEEE Trans. Speech Audio Process.*, vol. 7, no. 5, pp. 495-509, Sep. 1999.
- [9] J. Samuelsson and P. Hedelin, "Recursive coding of spectrum parameters," *IEEE Trans. Speech Audio Process.*, vol. 9, no. 5, pp. 492-503, July 2001.
- [10] T. Eriksson and F. Nordén, "Memory-based vector quantization of LSF parameters by a power series approximation," *IEEE Trans. Audio, Speech, Language Process.*, vol. 15, no. 4, pp. 1146-1155, May 2007.
- [11] Y. Wang and Q.-F. Zhu, "Error control and concealment for video communication: a review," *Proc. IEEE*, vol. 86, no. 5, pp. 974-997, May 1998.
- [12] J. Chakareski, S. Han, and B. Girod, "Layered coding vs. multiple descriptions for video streaming over multiple paths," in *Proc. 11th ACM Int. Conf. Multimedia*, 2003, pp. 422-431.
- [13] C. Perkins, O. Hodson, and V. Hardman, "A survey of packet loss recovery techniques for streaming audio," *IEEE Network*, vol. 12, no. 5, pp. 40-48, Sep. 1998.
- [14] T. Cover and J. Thomas, *Elements of Information Theory*. New York: John Wiley & Sons, 1991.
- [15] A. A. E. Gamal and T. M. Cover, "Achievable rates for multiple descriptions," *IEEE Trans. Inf. Theory*, vol. 28, pp. 851-857, Nov. 1982.
- [16] L. Ozarow, "On a source coding problem with two channels and three receivers," *Bell Syst. Tech. J.*, vol. 59, pp. 1909-1921, Dec. 1980.
- [17] Z. Zhang and T. Berger, "New results in binary multiple descriptions," *IEEE Trans. Inf. Theory*, vol. 33, no. 4, pp. 502-521, July 1987.
- [18] R. Zamir, "Shannon-type bounds for multiple descriptions of a stationary source," *J. Comb., Inf., Syst. Sciences*, Dec. 2000.
- [19] V. Vaishampayan, "Design of multiple description scalar quantizers," *IEEE Trans. Inf. Theory*, vol. 39, no. 3, pp. 821-834, May 1993.
- [20] V. Vaishampayan and J. Domaszewicz, "Design of entropy-constrained multiple-description scalar quantizers," *IEEE Trans. Inf. Theory*, vol. 40, no. 1, pp. 245-250, Jan. 1994.
- [21] V. Vaishampayan, "Vector quantizer design for diversity systems," in *Proc. 25th Ann. Conf. Inform. Sci. Syst.*, Johns Hopkins University, Mar. 1991, pp. 564-569.
- [22] Y. Wang, M. Orchard, and A. Reibman, "Multiple description image coding for noisy channels by pairing transform coefficients," in *Proc. IEEE First Workshop Multimedia Signal Process.*, June 1997, pp. 419-424.
- [23] M. Orchard, Y. Wang, V. Vaishampayan, and A. Reibman, "Redundancy rate-distortion analysis of multiple description coding using pairwise correlating transforms," in *Proc. ICIP*, Oct. 1997, pp. 608-611.
- [24] Y. Wang, M. T. Orchard, V. Vaishampayan, and A. R. Reibman, "Multiple description coding using pairwise correlating transforms," *IEEE Trans. Image Process.*, vol. 10, pp. 351-366, 2001.
- [25] Y. Wang, A. Reibman, M. Orchard, and H. Jafarkhani, "An improvement to multiple description transform coding," *IEEE Trans. Signal Process.*, vol. 50, no. 11, pp. 2843-2854, Nov. 2002.
- [26] S. Servetto, V. A. Vaishampayan, and N. J. A. Sloane, "Multiple description lattice vector quantization," in *Proc. Data Compression Conf.*, Mar. 1999, pp. 13-22.
- [27] S. Diggavi, N. Sloane, and V. Vaishampayan, "Design of asymmetric multiple description lattice vector quantizers," in *Proc. Data Compression Conf.*, 2000, pp. 490-499.
- [28] V. A. Vaishampayan, N. J. A. Sloane, and S. D. Servetto, "Multiple-description vector quantization with lattice codebooks: design and analysis," *IEEE Trans. Inf. Theory*, vol. 47, no. 5, pp. 1718-1734, July 2001.
- [29] P. L. Dragotti, S. D. Servetto, and M. Vetterli, "Optimal filter banks for multiple description coding: analysis and synthesis," *IEEE Trans. Inf. Theory*, vol. 48, pp. 323-332, 2002.
- [30] V. Vaishampayan, J.-C. Batllo, and A. R. Calderbank, "On reducing granular distortion in multiple description quantization," in *Proc. ISIT*, Aug. 1998, p. 98.
- [31] H. Jafarkhani and V. Tarokh, "Multiple description trellis coded quantization," in *Proc. ICIP*, Oct. 1998, pp. 669-673, vol. 1.
- [32] —, "Multiple description trellis-coded quantization," *IEEE Trans. Commun.*, vol. 47, no. 6, pp. 799-803, June 1999.
- [33] N. S. Jayant, "Sub-sampling of a DPCM speech channel to provide two self-contained half-rate channels," *Bell Syst. Tech. J.*, vol. 60, pp. 501-509, Apr. 1989.
- [34] A. Ingle and V. Vaishampayan, "DPCM system design for diversity systems with applications to packetized speech," *IEEE Trans. Speech Audio Process.*, vol. 3, no. 1, pp. 48-58, Jan. 1995.
- [35] P. Yahampath and P. Rondeau, "Multiple-description predictive-vector quantization with applications to low bit-rate speech coding over networks," vol. 15, no. 3, pp. 749-755, Mar. 2007.
- [36] "ITU-T recommendation G.726 40, 32, 24, 16 kbit/s adaptive differential pulse code modulation (adpcm)," ITU-T, 1990.
- [37] A. Tamhankar and K. Rao, "An overview of H.264/MPEG-4 part 10," in *Proc. 4th EURASIP Conf. Video/Image Process. Multimedia Commun.*, July 2003, pp. 1-51, vol. 1.
- [38] F. Itakura, "Line spectrum representation of linear predictive coefficients," *J. Acoust. Soc. Amer.*, vol. 57, no. 1, p. S35, 1975.
- [39] D. Persson and T. Eriksson, "Power series quantization for noisy channels," *IEEE Trans. Commun.*, vol. 58, pp. 1405-1414, 2010.
- [40] K. K. Paliwal and W. B. Kleijn, "Quantization of LPC parameters," in *Speech Coding and Synthesis*, W. Kleijn and K. Paliwal, editors. Amsterdam, The Netherlands: Elsevier, 1995, pp. 433-466.
- [41] P. Yahampath, "On index assignment and the design of multiple description quantizers," in *Proc. ICASSP*, May 2004, pp. 597-600, vol. 4.

Daniel Persson was born in Halmstad, Sweden, in 1977. He graduated from Ecole Polytechnique, Paris, France, in 2002, received the M.Sc. degree in engineering physics from Chalmers University of Technology, Göteborg, Sweden, in 2002, and the Ph.D. degree in electrical engineering from Chalmers University of Technology in 2009. He is currently a postdoctoral researcher in the Communication Systems Division, Linköping University, Sweden. His research interests are joint source-channel coding and MIMO detection.

Thomas Eriksson was born in Skövde, Sweden, on April 7, 1964. He received the M.Sc. degree in electrical engineering and the Ph.D. degree in information theory from Chalmers University of Technology, Göteborg, Sweden, in 1990 and 1996, respectively.

He was at AT&T Labs-Research from 1997 to 1998, and in 1998 and 1999, he was working on a joint research project with the Royal Institute of Technology and Ericsson Radio Systems AB.

Since 1999, he has been an Associate Professor at Chalmers University of Technology, and his research interests include vector quantization, speaker recognition, and system modeling of nonideal hardware.