

Turbo Trellis-Coded Quantization

Vivien Chappelier, Christine Guillemot and Slavica Marinković

IRISA/Université de Rennes 1, Campus de Beaulieu, 35042 RENNES Cedex FRANCE

Mailto: firstname.lastname@irisa.fr

Abstract: *This paper describes a new quantization technique based on the application of turbo coding principles to trellis-coded quantization (TCQ). The quantizer structure is described and a turbo quantization algorithm is presented. Turbo TCQ is used to encode a uniform i.i.d. source and compared to TCQ and scalar quantization in terms of rate-distortion performance. Convergence issues are addressed and a technique using multiple labellings of the trellis is presented to improve the performance. An extension to vector quantization is also presented.*

Keywords: Quantization, Trellis coding, Turbo codes

1. INTRODUCTION

The discovery of turbo codes by Berrou *et al.* [1] has led to substantial improvements in the field of channel coding, by providing efficient error correcting codes with low computational complexity. These codes are composed of two systematic convolutional component codes put in parallel. An interleaver, known to both the encoder and decoder, scrambles the input bitstream of one of the component coders. The turbo coded output consists of the systematic bits, and the redundancy bits of both coders. The decoding of a turbo code is performed by soft-output decoding algorithm applied iteratively on two trellises of the component code in parallel. An iteration consists in applying the maximum a posteriori [2] or soft-output Viterbi algorithm on the trellis of one decoder, with a priori information on the bits to decode, which is then subtracted from the soft-output to get the extrinsic information. The extrinsic information of one decoder is sent to the other trellis through the interleaver to be used as a priori information for the next decoding iteration. After a few iterations, the soft output converges and is thresholded to get the decoded message.

Turbo codes have been successfully combined with Ungerboeck's trellis-coded modulation (TCM) [3] to provide very efficient modulation schemes. In TCM, the constellation of a 2^n -state modulation is doubled and the output of a binary convolutional encoder of

rate $\frac{n}{n+1}$ is used to select one of the 2^{n+1} states. In [4], Benedetto *et al.* propose to use a turbo code instead of the convolutional code in TCM, achieving unprecedented error rates of 10^{-7} within 1 dB from the Shannon limit. In order to obtain a rate of $\frac{n}{n+1}$, the systematic bits of the turbo code are punctured. Robertson and Wörz [5] propose a similar scheme in which the parity bits are punctured instead.

Trellis-coded Quantization (TCQ), introduced by Marcellin and Fisher [6], applies the set-partitioning and trellis-coding principles of Ungerboeck to scalar quantization to achieve substantially better coding performance over Lloyd-Max scalar quantization at a low computational cost. For uniform memoryless sources, 256-state TCQ on 1000 samples even outperforms vector quantization based on the best known lattices up to dimension 24, and achieves a distortion within 0.21dB of the distortion-rate function. TCQ was further extended to trellis-coded vector quantization (TCVQ) by applying the set-partitioning on vector quantization [7].

As TCQ is very similar in principle to TCM and has proved to be an efficient quantization technique, it is interesting to examine the possibility of extending it to a turbo code based TCQ. In this paper we present a new turbo trellis-coded quantization scheme (TTCQ) and compare it to scalar quantization, TCQ, and to the distortion-rate bound. We focus on uniform memoryless sources for sake of simplicity. Moreover using a uniform quantizer on non-uniform sources is relevant when followed by an entropy coder. Despite issues with respect to the convergence of the turbo code, TTCQ outperforms TCQ by 0.2dB for medium-sized sequences.

We further apply our turbo quantizing method to vector quantization and compare it in terms of convergence and rate-distortion performance to the other considered quantizers. Finally, we propose a method to improve the performance of the presented quantizers by selecting the best one among multiple labellings of the trellis.

2. TCQ

TCQ is a particular kind of vector quantization of the dimension of the sequence length. It aims at

reaching the rate-distortion bound for vector quantizers[6] by partitioning a multidimensional grid into cosets by means of a convolutional code. The latter is chosen so that the coset lattices show good euclidian distance properties.

The quantizer is very similar to a TCM decoder. The quantizing algorithm applies the Viterbi algorithm on the trellis of a convolutional code to find the best sequence of bits minimizing the quantization distortion.

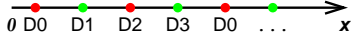


Figure 1: Set partitioning

Let $\mathbf{y} \in \mathbb{R}^N$ a vector to be quantized. For each component y_i , the reconstruction set of the uniform scalar quantizer on $n+1$ bits is partitioned in 4 codebooks of 2^{n-1} codewords. These codebooks are grouped in 2 cosets $D_0 \cup D_2$ and $D_1 \cup D_3$, only one of which can be used for quantization at a given instant i . A total of n bits per sample is transmitted. For each sample one bit is used to select a codebook inside the allowed coset whereas the $n - 1$ other bits are used to index a codeword within this codebook. The codebooks are chosen so that each coset is still a good quantizer for the source. For instance, UTCQ [8] performs well on gaussian and laplacian sources, by keeping the highly probable zero reconstruction value accessible in both cosets.

Each branch of a rate $\frac{1}{2}$ convolutional code trellis is labelled by a unique codebook D_k . The labelling defines N-dimensional vectors in the *super-codebook* of the vector quantization.

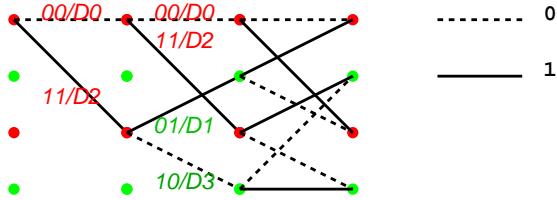


Figure 2: Trellis labelling

A trellis branch is assigned a metric ρ_i which corresponds to the distortion introduced by quantizing on the closest codeword in the codebook corresponding to that branch. The path with the minimum total distortion is found by applying the Viterbi algorithm. The convolutional code input sequence which generates this minimum distortion path determines the sequence of codebooks.

In the dequantizer the bits defining the minimum distortion path are encoded with the convolutional

code to recover the sequence of codebooks. The sample is reconstructed by indexing the codebook with the $n - 1$ bits for each instant i .

3. TURBO TCQ

3.1. Principle

The turbo trellis-coded quantizer is composed of two convolutional code trellises, constructed as in TCQ, with distortion metrics computed from the direct sequence for trellis \mathcal{A} , and from the interleaved sequence for trellis \mathcal{B} . As in turbo TCM [5], parity bits are punctured, i.e. branch metrics are zeroed for odd (resp. even) instants in trellis \mathcal{A} (resp. \mathcal{B}).

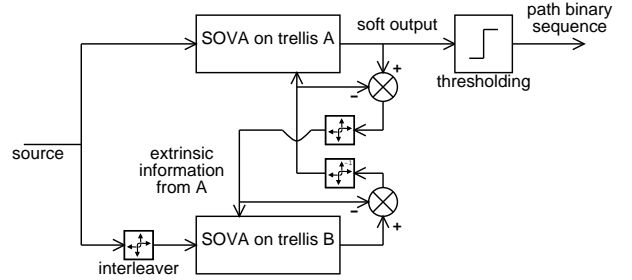


Figure 3: Quantizer

Quantization is performed iteratively. At each step, a soft output Viterbi algorithm runs on one of the trellises and outputs *a posteriori metrics*. A posteriori metric for each instant i is defined as the difference in distortion between the best path having a 1 at time i and the best path having a 0 at time i . The *a priori metrics*, initialized to 0, are subtracted from the *a posteriori metrics* to give the extrinsic information which is interleaved and fed as a priori metrics to the other trellis. This process is repeated until the soft output of the two trellises converge. In order to get the final binary output sequence, the soft output is thresholded. For each instant, the $n - 1$ bits are used to index the best codeword in the selected codebook.

We used an odd-even S-random interleaver, to keep odd (resp. even) samples at odd (resp. even) positions, and avoid small cycles in the turbo code graph.

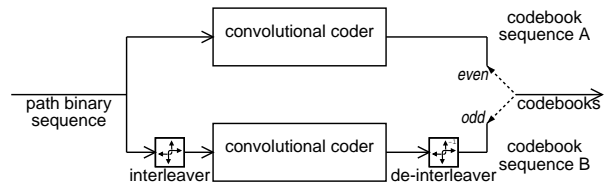


Figure 4: Dequantizer

The dequantizer consists in two convolutional coders, one reading the stream of bits directly and the other through the interleaver. Only one output is valid at a given time because the code has been punctured during quantization, i.e. even samples use the output of the direct coder whereas odd samples use the output of the interleaved coder. As in TCQ, the $n - 1$ bits are used to select the codeword inside the selected codebook to use as the reconstruction value.

3.2. Convergence issues

The main difference between a turbo TCM and turbo TCQ lies in the distribution of the random vector \mathbf{y} on which the iterative decoding (resp. quantizing) is applied. In the case of channel codes, the general assumption is that the encoder is emitting a codeword on which gaussian noise is added by the channel. The distribution of \mathbf{y} can be represented a sum of multidimensional gaussians centered on each codeword. Hence the probability of being far from a codeword is tending towards zero as the dimension of \mathbf{y} increases. Since typical sequence lengths are large, this probability can be approximated to zero.

However, in our case, assuming a uniform, gaussian or laplacian distribution of the components y_i ¹, this property does not hold. We cannot assume vector \mathbf{y} is initially close to a codeword of the turbo code. This leads to convergence problems, as the metrics do not favor one particular path with respect to others, unlike in the case of channel codes. Thus, the approximation that Pearl’s message passing algorithm [9] designed for trees can be applied on the turbo code cyclic graph does not work well, as suggested in [10].

Nevertheless, a significant improvement over TCQ for medium-sized test sequences can be obtained by keeping track of the output of each iteration. A tested sequence is said to have converged if its hard output is unchanged for more than 50 iterations. For inputs where the turbo code does not converge, the quantizer keeps the binary sequence minimizing the distortion among all the sequences it has tested.

Table 1 shows the average improvement over TCQ and shows that improvement is larger for the converging cases. In the next sections we introduce a relabelling technique and an extension to vector quantization aimed at enhancing TTCQ by improving convergence.

quantizer	TCQ	TTCQ [conv/no conv]
SNR (dB) 2bpp	12.39	12.64 [12.90/12.46]
SNR (dB) 6bpp	37.00	37.20 [37.34/37.10]

Table 1: 64-sample TTCQ at various rates, 10^4 iter.

¹which corresponds to most practical cases of quantization

3.3. Relabelling of the trellis

Since the convergence of the turbo code is highly dependent on a good initialization of the iterative process, we tried to quantize the input vector on two different super-codebooks differing only from the labelling of the trellis branches. As the super-codebooks are disjoint by construction, the input vector is closer to a codeword of one of the super-codebooks. Therefore, though a turbo TCQ on one of the super-codebooks may fail, it may converge on the other. Thus we do two turbo TCQ, one on each super-codebook, and keep the best output. However, one extra bit per sequence is need to tell the decoder which labelling to use. This technique is also extended to all four different possible labellings (QTTCQ). Another advantage of relabelling is that it improves quantization at the beginning of the sequence by limiting the impact of the choice of the initial state.

3.4. Vector quantization

In order to improve the performance of turbo TCQ for longer sequences, we investigated the possibility of using a vector quantizer instead of a scalar quantizer as the component quantizer. The input samples were grouped in 2D vectors to be quantized on one of two hexagonal grids. The grid allowed for quantization at a given instant is dependent on the choices made for the quantization of the previous samples, as in scalar turbo TCQ. Each grid corresponds to a coset and selecting a codebook in a grid determines the grid allowed at the next instant. Figure 5 illustrates this partitioning of the 2D space.

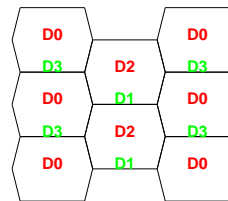


Figure 5: Partitioning in 2D

The algorithm used is exactly the same as in scalar turbo TCQ, except for the distortion metrics which are computed for a 2D vector of two input samples being quantized on an hexagonal grid.

4. RESULTS

The results presented here are an average of 10^4 experiments using a 5/7 systematic code. For medium-sized sequences of 64 samples, Table 1 shows an improvement of 0.25dB from TTCQ compared to TCQ with the same component code at a rate of 2 bits/sample. However, 10^4 iterations is large when compared to channel decoding, and for longer sequences

quantizer	single	dual	quad
scalar theoretical (dB)	48.16	-	-
rate-distortion bound	49.43	-	-
TCQ 4 states	49.06	49.11	49.13
TCQ 256 states	49.18	49.20	49.23
TTCQ 4 states, 10^3 iter.	49.16	49.22	49.27
TTCQ 4 states, 10^4 iter.	49.23	49.26	49.29

Table 2: 8 bit quantization on 50-sample sequences

the improvement is canceled by the decrease in the number of converging cases.

Table 2 shows that dual relabelling helps in obtaining similar results as the standard turbo TCQ, with less iterations. TCQ is also improved, by alleviating the constraint due to the selection of the initial state. Using four labelling increases the performance even more. The gain obtained from the relabelling method is also clearly appearing in figure 6, with QTTCQ achieving rate-distortion performance within 0.13dB of the bound for sequences of length 70 at 8 bits/sample. The results presented take into account the extra bit(s) needed to tell the decoder which super-codebook to use. For longer sequences, TTCVQ performs better than TTCQ due to better convergence properties, as shown in Table 3. Figure 6 suggests that using VQ translates the benefits of TTCQ in higher dimensions, though a fall in performance is still appearing past dimension 200.

5. CONCLUSIONS, FUTURE WORK

We have presented a new method using the turbo principles for quantization. Though the iterative process may fail to converge, a significant gain over existing lattice quantizers is obtained, for a memoryless uniform source and medium-sized sequences. Alternate labellings of the trellis allowed to improve the performance of the turbo quantizer. Though the computational complexity of the quantizer may be

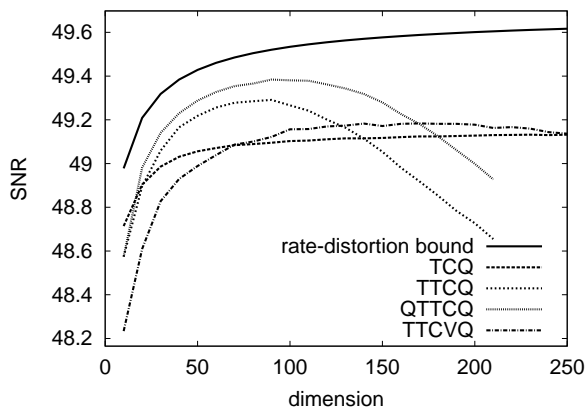


Figure 6: Distortion against sequence length at 8 bits/sample for various quantizers

quantizer	TCQ	TTCQ	TTCVQ
SNR 50 samples (dB)	49.05	49.16	48.99
convergence [†]		36.6%	36.0%
SNR 200 samples (dB)	49.13	48.31	49.17
convergence [†]		2.1%	13.4%

Table 3: convergence ratio for 10^3 iterations at 8 bpp

large, the dequantizer stays very simple, making the proposed approach suitable for asymmetric applications. Since TCQ has been used successfully for the quantization of gaussian and laplacian sources, the proposed scheme could be used on these sources too. Finally, increasing the dimension of the VQ used in TTCVQ could improve its performance on longer sequences.

REFERENCES

- [1] C. Berrou, A. Clavier, and P. Thitimajshima. Near shannon limit error-correcting coding: Turbo codes. In *IEEE Inter. Conf. Comm.*, pp. 1064–1070, May 1993.
- [2] L.R. Bahl, J. Cocke, F. Jelinek, and J. Raviv. Optimal decoding of linear codes for minimizing symbol error rate. *IEEE Trans. Inf. Theory*, IT-20:284–287, Mar. 1974.
- [3] G. Ungerboeck. Channel coding with multi-level/phase signals. *IEEE Trans. Inf. Theory*, IT-28(1):55–67, Jan. 1982.
- [4] S. Benedetto, D. Divsalar, G. Montorsi, and F. Pollara. Parallel concatenated trellis coded modulation. In *IEEE Inter. Conf. Comm.*, volume 2, pp. 974–978, 1996.
- [5] P. Robertson and T. Wörz. Bandwidth-efficient turbo trellis-coded modulation using punctured component codes. *IEEE Journal on Selected Areas on Communication*, 16(2):206–218, Feb. 1998.
- [6] M.W. Marcellin and T.R. Fisher. Trellis-coded quantization of memoryless and gauss-markov sources. *IEEE Trans. Comm.*, 38:82–93, Jan. 1990.
- [7] T.R. Fischer, M.W. Marcellin, and M. Wang. Trellis-coded vector quantization. *IEEE Trans. Inf. Theory*, IT-37:1551–1566, Nov. 1991.
- [8] J.H. Kasner, M.W. Marcellin, and B.R. Hunt. Universal trellis coded quantization. *submitted to IEEE Trans. Image Processing*.
- [9] J. Pearl. *Probabilistic Reasoning in Intelligent Systems, Networks of Plausible Inference*. Morgan Kaufmann Publishers, Inc., 1988.
- [10] K.P. Murphy, Y. Weiss, and M. Jordan. Loopy belief propagation for approximate inference: an empirical study. In *Uncertainty in AI*, volume 9, pp. 467–475, 1999.

[†]number of converging cases over all tested sequences