EURASIP Journal on Applied Signal Processing 2005:6, 906–927

Joint Source-Channel Decoding of Variable-Length Codes with Soft Information: A Survey

Christine Guillemot
IRISA-INRIA, Campus de Beaulieu, 35042 Rennes Cedex, France
Email: christine.guillemot@irisa.fr

Pierre Siohan
R&D Division, France Telecom, 35512 Rennes Cedex, France
Email: pierre.siohan@francetelecom.com

Received 13 October 2003; Revised 27 August 2004

Multimedia transmission over time-varying wireless channels presents a number of challenges beyond existing capabilities conceived so far for third-generation networks. Efficient quality-of-service (QoS) provisioning for multimedia on these channels may in particular require a loosening and a rethinking of the layer separation principle. In that context, joint source-channel decoding (JSCD) strategies have gained attention as viable alternatives to separate decoding of source and channel codes. A statistical framework based on hidden Markov models (HMMs) capturing dependencies between the source and channel coding components sets the foundation for optimal design of techniques of joint decoding of source and channel codes. The problem has been largely addressed in the research community, by considering both fixed-length codes (FLC) and variable-length source codes (VLC) widely used in compression standards. Joint source-channel decoding of VLC raises specific difficulties due to the fact that the segmentation of the received bitstream into source symbols is random. This paper makes a survey of recent theoretical and practical advances in the area of JSCD with soft information of VLC-encoded sources. It first describes the main paths followed for designing efficient estimators for VLC-encoded sources, the key component of the JSCD iterative structure. It then presents the main issues involved in the application of the turbo principle to JSCD of VLC-encoded sources as well as the main approaches to source-controlled channel decoding. This survey terminates by performance illustrations with real image and video decoding systems.

Keywords and phrases: joint source-channel decoding, source-controlled decoding, turbo principle, variable-length codes.

1. INTRODUCTION

The advent of wireless communications, ultimately in a global mobility context with highly varying channel characteristics, is creating challenging problems in the area of coding. Design principles prevailing so far and stemming from Shannon's source and channel separation theorem are being reconsidered. The separation theorem, stating that source and channel optimum performance bounds can be approached as close as desired by designing independently source and channel coding strategies, holds only under asymptotic conditions where both codes are allowed infinite length and complexity, and under conditions of source stationarity. If the design of the system is heavily constrained in terms of complexity or delay, source and channel coders can be largely suboptimal, leading to residual channel error rates, which can be large and lead to dramatic source symbol error rates. The assumption prevailing so far was essentially that the lower layers would offer a guaranteed delivery service, with a null residual bit error rate: for example, the error detection mechanism supported by the user datagram protocol (UDP) discards all UDP packets corrupted by bit errors, even if those errors are occurring in the packet payload. The specification of a version of UDP, called UDP-Lite [1], that would allow to pass erroneous data to the application layer (i.e., to the source decoder) to make the best use of error-resilient decoding systems is under study within the Internet Engineering Task Force (IETF).

These evolving trends have led to considering joint source-channel coding (JSCC) and decoding (JSCD) strategies as viable alternatives for reliable multimedia communication over noisy channels. Researchers have taken several paths toward the design of efficient JSCC and JSCD strategies, including the design of unequal error protection strategies [2], of channel optimized quantizers [3, 4], and of resilient entropy codes [5, 6]. Here, we focus on the JSCD problem in a classical communication chain based on standardized systems and making use of a source coder aiming

at decorrelating the signal followed by a channel coder that aims at reaugmenting the redundancy in the transmitted stream in order to cope with transmission errors. The key idea of JSCD is to exploit jointly the residual redundancy of the source-coded stream (i.e., exploiting the sub-optimality of the source coder) and the redundancy introduced by the channel code in order to correct bit errors, and find the best source-symbols estimates.

Early work reported in the literature assumed fixed-length representations (FLC) for the quantized-source indexes [7, 8, 9, 10]. Correlation between successive indexes in a Markovian framework is exploited to find maximum a posteriori (MAP) or minimum mean square error (MMSE) estimates. The applications targeted by research on error-resilient FLC decoding and JSCD of FLC are essentially speech applications making use for instance of CELP codecs. However, the wide use of VLC in data compression, in particular for compressing images and video signals, has motivated more recent consideration of variable-length coded streams. As in the case of FLC, VLC decoding with soft information amounts to capitalize on source coder suboptimality, by exploiting residual source redundancy (the so-called "excess-rate") [11, 12, 13, 14, 15]. However, VLC decoding raises extra difficulties resulting from the lack of synchronization between the symbol and bit instants in presence of bit errors. In other words, the position of the symbol boundaries in the sequence of noisy bits (or measurements) is not known with certainty. This position is indeed a random variable, the value of which depends on the realization of all the previous symbols in the sequence. Hence, the segmentation of the bitstream into codewords is random, which is not the case for FLCs. The problem becomes a joint problem of segmentation and estimation which is best solved by exploiting both intersymbol correlation (when the source is not white) as well as inner-codeword redundancy resulting from the entropy code suboptimality.

This problem has first been addressed by considering tree-based codes such as Huffman codes [11, 16, 17]. In this case, the entropy-coded bitstream can be modelled as a semi-Markov process, that is, as a function of a hidden Markov process. The resulting dependency structures are well adapted for MAP (maximum a posteriori) and MPM (maximum of posterior marginals) estimation, making use of soft-input soft-output dynamic decoding algorithms such as the soft-output Viterbi algorithm (SOVA) [18] or the BCJR algorithm [19]. To solve this problem, various trellis representations have been proposed, either assuming the source to be i.i.d. as in [20, 21], or also taking into account the intersymbol dependencies. In source coding, the mean square error (MSE) being a privileged performance measure, a conditional mean or MMSE criterion can also be used, possibly with approximations to maintain the computational complexity within a tractable range [22].

The introduction of arithmetic codes in practical systems (e.g., JPEG-2000, H.264) has then moved the effort towards the design of robust decoding techniques of arithmetic codes. Sequential decoding of arithmetic codes is investigated in [23] for supporting error correction capabilities. Sequential decoding with soft output and different paths pruning techniques are described in [24, 25]. Additional error detection and correction capabilities are obtained in [26] by reintroducing redundancy in the form of parity-check bits embedded in the arithmetic coding procedure. A probability interval not assigned to a symbol of the source alphabet or markers inserted at known positions in the sequence of symbols to be encoded is exploited for error detection in [27, 28, 29]. The authors in [30] consider quasiarithmetic codes which, in contrast with optimal arithmetic codes, can be modelled as finite-state automata (FSA).

When an error-correcting code (ECC) is present in the communication chain, optimum decoding can be achieved by making joint use of both forms of redundancy: the source "excess-rate" and the redundancy introduced by the ECC. This is the key idea underlying all joint source-channel decoding strategies. Joint use of correlation between quantized indexes (i.e., using fixed-length representations of the indexes) and of redundancy introduced by a channel turbo coder is proposed in [31]. The approach combines the Markovian source model with a parallel turbo coder model in a product model. In order to reduce the complexity, an iterative structure, in the spirit of serial turbo codes where the source coder is separated from the channel coder by an interleaver, is described in [32]. The convergence behavior of iterative source-channel decoding with fixed-length source codes and a serial structure is studied in [33] using EXIT charts [34]. The gain brought by the iterations is obviously very much dependent on the amount of correlation present on both sides of the interleaver.

Models incorporating both VLC-encoded sources and channel codes have been considered in [16, 17, 35, 36]. The authors in [16] derive a global stochastic automaton model of the transmitted bitstream by computing the product of the separate models for the Markov source, the source coder, and the channel coder. The resulting automaton is used to perform a MAP decoding with the Viterbi algorithm. The approach provides optimal joint decoding of the chain, but remains untractable for realistic applications because of state explosion. In [35, 36, 37], the authors remove the memory assumption for the source. They propose a turbo-like iterative decoder for estimating the transmitted symbol stream, which alternates channel decoding and VLC decoding. This solution has the advantage of using one model at a time, thus avoiding the state explosion phenomenon. The authors in [14] push further the above idea by designing an iterative estimation technique alternating the use of the three models (Markov source, source coder, and channel coder). A parallel iterative joint source-channel decoding structure is also proposed in [38].

Alternatively, the a priori source statistic information can be directly taken into account in the channel decoder by designing source-controlled channel decoding approaches. Initially proposed in [39], the approach has been mostly investigated in the case where a source coder using FLC is used in conjunction with a convolutional channel coder [39], or with a turbo channel coder [40, 41]. This approach, introduced at first with fixed-length codes (FLCs), has been extended to

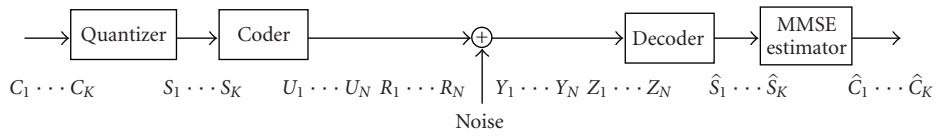


Figure 1: Overview of the source-channel coding chain.

cover the case of VLC used in conjunction with convolutional and turbo codes [42, 43].

The rest of the paper is organized as follows. Section 2 describes part of the notations used and briefly revisits estimation criteria (MAP, MPM, MMSE) and algorithms on which we rely in the sequel. Section 3 presents models of dependencies and corresponding graph representations for source coders which can be modelled as FSA. Estimation using trellis decoding algorithms can be run on the resulting dependency graphs. When the coder cannot be modelled as a finite-state automaton, sequential decoding with soft output can be applied as explained in Section 3.5. The problem of complexity resulting from large state-space dimensions for realistic sources is addressed in Section 3.6 where the graph of dependencies of the source coder is separated into two separate graphs exchanging information along a dependency tree-structure. Mechanisms to improve the decoder's resynchronization capability are described in Section 4. Section 5 presents joint source-channel decoding principles placing the above material in an iterative decoding structure in the spirit of serial and parallel turbo codes. Section 6 gives an overview of channel decoding with the help of a priori source information. In Section 7, we present the estimation problem of the source statistics from noisy information. Finally, Section 8, in light of performance illustrations with real image and video coding/decoding systems, discusses the potential of JSCD strategies.

2. BACKGROUND

In order to introduce the notations and the background of estimation techniques, we first consider the simple coding and transmission chain depicted in Figure 1. We reserve capital letters to random variables, and small letters to values of these variables. Let $C_1^K = C_1 \cdots C_K$ be a sequence of source symbols to be quantized and coded and let $S_1^K = S_1 \cdots S_K$ be the corresponding sequence of quantized symbols taking their values in a finite alphabet $\mathcal{A}$ composed of $Q = 2^q$ symbols, $\mathcal{A} = \{a_1, a_2, \ldots, a_i, \ldots, a_Q\}$. The sequence S is then coded into a sequence of bits $U_1^N = U_1 \cdots U_N$, by means of a coder, that can encompass a source or/and a channel coder as we will see later.

The length N of the useful bitstream is a random variable, a function of S_1^K and of the coding processes involved in the chain. However, in most transmission systems, the encoded version of S_1^K is delimited by a prefix and a suffix that allow a reliable isolation of U_1^N. Therefore, one can assume that N is perfectly observed. A sequence of redundant bits $R_1^N = R_1 \cdots R_N$ may be added to U_1^N by means of a correcting code (Figure 1 and the notation R_1^N correspond to the case where a systematic rate 1/2 code is used).

The bitstream U_1^N is sent over a memoryless channel and received as measurements Y_1^N (see Figure 1). Let $Y_1^N = Y_1 \cdots Y_N$ be pointwise (noisy) measurements on the sequence of bits U_1^N. The sequence Y_1^N models the output of the discrete channel, the quantities $y_1^N = y_1 \cdots y_N$ denoting the particular values of Y_1^N observed at the output of the channel. The decoding problem consists in finding an estimate $\hat{S}_1^K$ of the sequence S_1^K, and ultimately reconstructing C_1^K, given the observed values y_1^N on the useful bits and z_1^N on the redundant bits.

Assuming that C_1^K and S_1^K are standard Markov processes, the problem becomes a classical hidden Markov inference problem for which efficient algorithms, known under a variety of names (e.g., Kalman smoother, Raugh-Tung-Striebel algorithm, BCJR algorithm, belief propagation, sum-product algorithm, etc.), exist. The problem is indeed to estimate the sequence of hidden states of a Markovian source through observations of the output of a memoryless channel. The estimation algorithms often differ by the estimation criteria (MAP, maximum likelihood (ML), MPM, and minimum mean square error (MMSE)) and the way the computations are organized. The decision rules can be either optimum with respect to a sequence of symbols or with respect to individual symbols.

2.1. MAP estimates (or "sequence" MAP decoding)

The MAP estimate of the whole process S_1^K based on all available measurements Y_1^N can be expressed as[1]

$$\hat{S}_1^K = \hat{S}_1 \cdots \hat{S}_K = \arg\max_{s_1 \cdots s_K} \mathbb{P}(s_1 \cdots s_K | y_1 \cdots y_N). \quad (1)$$

The optimization is thus made over all possible *sequences* realizations s_1^K of bit length N. Assuming that the symbols S_k are converted into a fixed number of bits (using a FLC), when the a priori information is equally distributed, it can be shown easily that, using the Bayes formula, the "sequence" MAP ($\mathbb{P}(S_1^K | Y_1^N)$) is equivalent to the ML estimate ($\mathbb{P}(Y_1^N | S_1^K)$). The ML estimate can be computed with the Viterbi algorithm (VA). When the a priori information is not equally distributed, the "sequence" MAP can still be derived from the ML estimate, as a subproduct of the VA, if the metric incorporates the a priori probabilities $\mathbb{P}(S_{k+1} | S_k)$. If the

[1] For notational convenience, in the sections where a channel coder is not explicitly involved, the channel input and output are denoted by U_1^N and Y_1^N, respectively.

symbols S_k are converted into variable numbers of bits, one has to use instead the generalized Viterbi algorithm [44].

In ML and MAP estimations, the ratios of probabilities of trellis paths leading to a given state X to the sum of probabilities of all possible paths terminated at X are often computed as *likelihood ratios* or in the logarithmic domain as *log-likelihood ratios (LLRs)*. Modifications have been introduced in the VA in order to obtain at the decoder output, in addition to the "hard"-decoded symbols, reliability information, leading to the soft-output Viterbi algorithm (SOVA) [18]. The ML or sequence MAP estimation algorithms do supply sequence a posteriori probabilities but not actual a posteriori symbol probabilities, hence are not optimal in the sense of symbol error probability.

2.2. MPM estimation (or "symbol-by-symbol" MAP decoding)

The symbol-by-symbol MAP decoding algorithms search for the MPM estimates, that is, estimate each hidden state of the Markov chain individually according to

$$\hat{S}_k = \arg\max_{s_k} \mathbb{P}\left(S_k = s_k \mid Y_1^N = y_1^N\right). \tag{2}$$

Assuming that the symbols S_k are converted into a fixed number of bits (using a FLC), computations can be organized around the factorization

$$\mathbb{P}\left(S_k | Y_1^N\right) \propto \mathbb{P}\left(S_k, Y_1^n\right) \cdot \mathbb{P}\left(Y_{n+1}^N | S_k\right), \tag{3}$$

where $\propto$ denotes a renormalization. The measures Y_n on bits U_n can indeed be converted into measures on symbols in a very straightforward manner. In the case of VLC encoding of the symbols S_k, the conversion brings some slight technical difficulty (we return to this point in Section 3).

First "symbol-by-symbol" MAP decoding algorithms have been known from the late sixties [45], early seventies [19, 46]. The Markov property allows a recursive computation of both terms of the right-hand side organized in a forward and backward recursion. The BCJR algorithm is a two-recursion algorithm involving soft decisions and estimating per-symbol a posteriori probabilities. To reconstruct the data sequence, the soft output of the BCJR algorithm are hard limited. The estimates need not form a connected path in the estimation trellis.

Because of its complexity, the implementation of the MAP estimation has been proposed in the logarithmic domain leading to a log-MAP algorithm [47, 48]. In its logarithmic form, exponentials related to the classical additive white Gaussian noise (AWGN) channel disappear, and multiplications become additions. Further simplifications and approximations to the log-MAP algorithm have been proposed in order to avoid calculating the actual probabilities. These simplifications consist in replacing the additions by some sort of MAX operations plus a logarithmic term $(\ln(1 + \exp(-|a - b|)))$. Ignoring the logarithmic term leads to the suboptimal variant known as the Max-Log-MAP algorithm [48].

2.3. MMSE decoding

The performance measure of source coding-decoding systems is traditionally the MSE between the reconstructed and the original signal. In that case, the MAP criterion is suboptimal. Optimal decoding is given instead by conditional mean or MMSE estimators. The decoder seeks the sequence of reconstruction values $(\hat{a}_1 \cdots \hat{a}_k \cdots \hat{a}_K)$, $\hat{a}_k \in \mathbb{R}$, $k = 1, \ldots, K$, for the sequence C_1^K. The values $\hat{a}_k$ may not belong to the alphabet used initially to quantize the sequence of symbols C_1^K. This sequence of reconstruction values should be such that the expected distortion on the reconstructed sequence C_1^K, given the sequence of observations Y_1^N, and denoted by $E[D(\hat{C}_1^K, C_1^K) | Y_1^N]$, is minimized. This expected distortion can be computed from the a posteriori probabilities (APPs) of the estimated quantized sequence $\hat{S}_1^K$, given the sequence of measurements, obtained as a result of the sequence MAP estimation described above.

However, minimizing $E[D(\hat{C}_1^K, C_1^K) | Y_1^N]$, that is, given the entire sequence of measurements, becomes rapidly untractable except in trivial cases. Approximate solutions (approximate MMSE estimators (AMMSE)) considering the expected distortion for each reconstructed symbol $E[D(\hat{C}_k, C_k) | Y_1^N]$ are used instead [22]. The problem then amounts to minimizing

$$\hat{D} = \sum_{k=1}^{K} \sum_{l=1}^{M} ||a_l - \hat{a}_k||^2 \mathbb{P}\left[\hat{S}_k = a_l \mid Y_1^N = y_1^N\right], \tag{4}$$

where M is the size of the reconstruction alphabet, and the reconstruction values $\hat{a}_k$ are centroids computed as

$$\hat{a}_k = \sum_{l=1}^{M} a_l \mathbb{P}\left[\hat{S}_k = a_l \mid Y_1^N = y_1^N\right]. \tag{5}$$

The term $\mathbb{P}[\hat{S}_k = a_l \mid Y_1^N = y_1^N]$ turns out to be the posterior marginals computed with the MPM strategy described above, that is, with the forward/backward recursion as in [19].

3. SOFT-INPUT SOFT-OUTPUT SOURCE DECODING

The application of the turbo principle to JSCD, according to serial or parallel structures, as we will see in Section 5, requires first to design soft-input soft-output source decoding algorithm. The problem is, given the sequence of observations Y_1^N (sequence of noisy bits received by the source decoder), to estimate the sequence of symbols S_1^K. The term "soft" here means that the decoder takes in input, and supplies, not only binary ("hard") decisions, but also a measure of confidence (i.e., a probability) on the bits. The dependencies between the quantized indexes are assumed to be Markovian, that is, the sequence of quantized indexes S_1^K is assumed to be a first-order Markov chain driven by conditional probabilities $\mathbb{P}(S_k | S_{k-1})$ and by initial stationary probabilities $\mathbb{P}(S_k)$. The Markovian assumption allows to represent the source as a FSA with a well-established graph representation, as shown in Figure 2a. The hidden states are

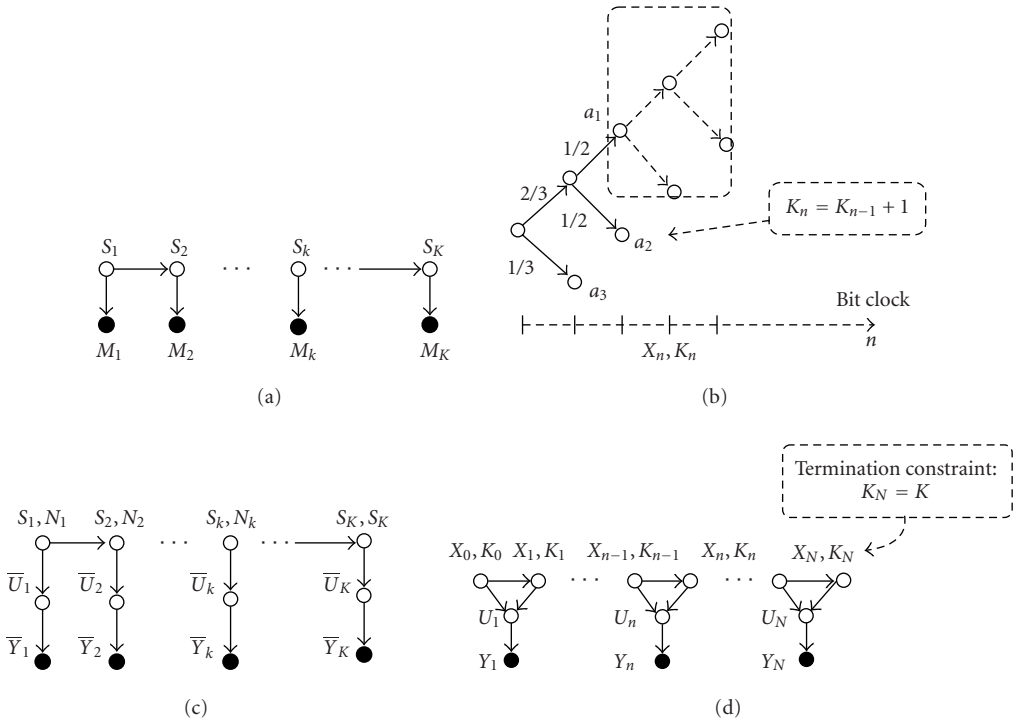


Figure 2: Graphical representation of source and of source-coder dependencies: (a) Markov source; (b) source HMM augmented with a counter N_k of the number of bits emitted at the symbol instant k; (c) example of codetree, the transition probabilities are written next to the branches; (d) coder HMM augmented with a counter K_n of the number of symbols encoded at instant n.

represented by nodes S_k, while transitions between states are represented by directed edges. In this general statement of the problem, we denote by M_k the set of observations on the hidden states S_k.

The design of soft-input soft-output source decoding algorithms requires first to construct models capturing the dependencies between the different variables representing the source of symbols and the coding process. The modelling of the dependencies between the variables involved in the coding chain can be performed by means of the Bayesian network formalism [49]. Bayesian networks are a natural tool to analyze the structure of stochastic dependencies and of constraints between variables, through graphical representations which provide the structures on which can be run the MAP or MMSE estimators.

3.1. Sources coded with fixed-length codes

The use of fixed-length source representations makes the problem much simpler: in the case of FLC, the segmentation of the received bitstream into symbol measurements is known. Symbols S_k are indeed translated into codewords $U_{(k-1)q+1}^{kq}$, where q is the length of the quantized source codewords, by a deterministic function. The set of observations M_k (see Figure 2a) is thus obtained by gathering the measurements $Y_{(k-1)q+1}^{kq}$. Estimation algorithms on the resulting symbol-trellis representation are thus readily available with complexity in $O(K \times |Q|^2)$, where Q is the size of the source alphabet. This approach has been followed in [7, 8, 9, 10] for source symbol estimation. One can alternatively consider a bit-trellis representation of the dependencies between the different variables, by noticing that estimating S_1^K is equivalent to estimating U_1^N and by regarding the decision tree generating the fixed length codewords $U_{(k-1)q+1}^{kq}$ as a finite-state stochastic automaton. Although, this bit-trellis representation is not of strong interest in the case of FLC, it is very useful for VLC to help with the bitstream segmentation problem. The approach is detailed below.

3.2. Sources coded with variable-length codes

We first consider the case of sources coded with binary variable-length codetrees, for example, Huffman [50] or reversible variable-length codes (RVLC) [51]. The difficulty inherent in the problem of soft decoding of variable-length coded sources is the lack of synchronization between the received bits and the symbols. The problem is hence to relate measurements, or subsets of the sequence of observations, to given symbols S_k. The positions of the symbol boundaries in the bitstream may not be estimated properly. The problem is hence a joint problem of segmentation (i.e., recovering symbol boundaries) and estimation. This problem can be addressed by regarding the coding and decoding processes as FSA modelling the output bitstream distribution. This is better explained in a simple example. We consider the simple source coder based on the binary codetree shown in Figure 2b. The example of Figure 2b is a Huffman tree

corresponding to a probability vector $\mathbb{P} = [1/3\ 1/3\ 1/3]$ and to the code $(00, 01, 1)$ for (a_1, a_2, a_3).

We first assume that the input of the coder is a *white* source. For this type of code, the encoding of a symbol[2] determines the choice of vertices in the binary decision tree. The decision tree can be regarded as a stochastic automaton that models the bitstream distribution. Each node ν of the tree identifies a state of the coder. Leaves of the tree represent terminated symbols, and are thus identified with the root of the tree, to prepare the production of another symbol code. The coder/decoder states can thus be defined by variables $X_n = (\nu)$, where ν is the index of an internal node of the tree. Successive branchings in the tree, hence transitions on the automaton, follow the source stationary distribution $\mathbb{P}$ and trigger the emission of the bits. This model leads naturally to a *bit-trellis* structure such as that used in [20, 21, 52, 53].

We now assume that the input of the coder is a Markov process. Optimal decoding requires to capture both inner codeword and intersymbol correlation, that is, the dependencies introduced by both the symbol source and the coder. In order to do so, in the model described above, one must in addition keep track of the last symbol produced, that is, connect the "local" models for the conditional distribution $\mathbb{P}(S_k|S_{k-1})$. In the case of an optimal coder, the value of the last symbol produced determines which codetree to use to code the next symbol. In practice, the same codetree is used for the subsequent symbols and the value of the last symbol produced thus determines which probabilities to use on the tree. The state of the automaton thus becomes $X_n = (\nu, s)$, where s is the memory of the last symbol produced. This connection of *local* automata to model the entire bitstream distribution amounts to identifying leaves of the tree with the root of the next tree as shown in Figure 2b. Successive branchings on the resulting tree thus follow the distribution of the source $\mathbb{P}(S_k|S_{k-1})$. Let X_n denote the state of the resulting automaton after n bits have been produced. The sequence $X_0, \ldots, X_N$ is therefore a Markov chain, and the output of the coder, function of transitions of this chain, that is, $U_n = \phi(X_{n-1}, X_n)$ can also be modelled as a function of a HMM graphically depicted in Figure 2d. The a posteriori probabilities on the bits $U_n = \phi(X_{n-1}, X_n)$ can thus be obtained by running a sequence MAP estimation (e.g., with a SOVA) or a symbol-by-symbol MAP estimation (e.g., with a BCJR algorithm) on the HMM defined by the pair (X_{n-1}, X_n). This model once more leads naturally to a *bit-trellis* structure [14], but in comparison with the case of memoryless sources, the state-space dimension is multiplied by the size of the source alphabet (corresponding to the number of leaves in the codetree), hence can be very high. The authors in [54] extend the bit trellis described in [21] to correlated sources and introduce a reduced structure with complete and incomplete states corresponding to leaf and intermediate nodes in the codetree. The corresponding complexity reduction induces some suboptimality.

[2]This can be extended in a straightforward way to blocks of l symbols taking their values in the product alphabet $\mathcal{A}^l$.

To help in selecting the right transition probability on symbols, that is, in segmenting the bitstream into codewords, the state variable can be augmented with a random variable K_n defined as a symbol counter $K_n = l$. Transitions on ν follow the branches of the tree determined by s, and s, l change each time one new symbol is produced. Since the transitions *probabilities* on the tree depend on s, one has to map $\mathbb{P}(s'|s)$ on the corresponding tree to determine $\mathbb{P}(\nu', s', l'|\nu, s, l)$. This leads to the augmented HMM defined by the pair of variables (X_n, K_n) and depicted in Figure 2d. Note that the symbol counter K_n helps selecting the right transition probability on symbols. So, when the source is a stationary Markov source, K_n becomes useless and can be removed. If the length of the symbol sequence is known, this information can be incorporated as a termination constraint (constraining the value of K_N) in order to help the decoder to resynchronize at the end of the sequence. All paths which do not correspond to the right number of symbols can then be eliminated. The use of the symbol counter leads to optimum decoding, however at the expense of a significant increase of the state-space dimension and of complexity.

Intersymbol correlation can also be naturally captured on a *symbol-trellis* structure [14, 35, 37]. A state in this model corresponds to a symbol S_k and to a random number of bits N_k produced at the symbol instant k, as shown in Figure 2c. If the number of transmitted symbols is known, an estimation algorithm based on this symbol clock model would yield an optimal sequence of pairs (S_k, N_k), that is, the best sequence of K symbols regardless of its length in number of bits. Knowledge on the number of bits can be incorporated as a constraint on the last pair (S_K, N_K), stating that N_K equals the required number of bits N. When the number of bits is known, and the number of symbols is left free, the Markov model on process $(S_k, N_k)_{k=1,\ldots,K}$ must be modified. First, K must be large enough to allow all symbol sequences of N bits. Then, once N_k reaches the required length, the model must enter and remain in a special state for which all future measurements are noninformative.

When both the numbers of symbols and of bits transmitted are known and used in the estimation, the two models lead to optimum decoding with the same complexity. However, in practice, the length of the bit sequence is naturally obtained from the bitstream structure and the corresponding syntax (e.g., markers). The information on the number of symbols would in many cases need to be transmitted. Note also that a section of the symbol trellis corresponds to a random number of observations. Efficient pruning then becomes more difficult: pruning techniques should indeed optimally compare probabilities derived from the same (and same number of) measures. Pruning techniques on bit trellises are then closer to optimum decoding. This explains why bit trellises have been the most widely used so far, with variants depending on the source model (memoryless [20, 21, 52, 53] or with memory [14, 54]), and on the side information required in the decoding, that is, knowledge of the number of transmitted bits [52], or of both transmitted bits and transmitted symbols [14, 35].

3.3. Sources coded with (quasi-) arithmetic codes

Soft-input soft-output decoding of arithmetically coded sources brings additional difficulties. An optimal arithmetic coder operates fractional subdivisions of the interval $[\text{low}, \text{up})$ (with low and up initialized to 0 and 1, resp.) according to the probabilities and cumulative probabilities of the source [55]. The coding process follows a Q-ary decision tree (for an alphabet of dimension Q) which can still be regarded as an automaton, however with a number of states growing exponentially with the number of symbols to be encoded. In addition, transitions to a given state depend on all the previous states. In the case of arithmetic coding, a direct application of the SOVA and BCJR algorithms would then be untractable. One has to rely instead on sequential decoding applied on the corresponding decision trees. We come back to this point in Section 3.5.

Let us for the time being consider a reduced precision implementation of arithmetic coding, also referred to as quasiarithmetic (QA) coding [56], which can be modelled as FSA. The QA coder operates integer subdivisions of an integer interval $[0, T)$. These integer interval subdivisions lead obviously to an approximation of the source distribution. The tradeoff between the state-space dimension and the source distribution approximation is controlled by the parameter T. It has been shown in [57] that, for a binary source, the variable T can be limited to a small value (down to 4) at a small cost in terms of compression. The strong advantage of quasiarithmetic coding versus arithmetic coding is that all states, state transitions, and outputs can be precomputed, thus allowing to first decouple the coding process from the source model, and second to construct a finite-state automaton. Hence, the models turn out to be naturally a product of the source and of the coder/decoder models. Details can be found in [30].

The QA decoding process can then be seen as following a binary decision tree, on which transitions are triggered by the received QA-coded bits. The states of the corresponding automaton are defined by two intervals: $[\text{low}\, U_n, \text{up}\, U_n)$ and $[\text{low}\, S_{K_n}, \text{up}\, S_{K_n})$. The interval $[\text{low}\, U_n, \text{up}\, U_n)$ defines the segment of the interval $[0, T)$ selected by a given input bit sequence U_1^n. The interval $[\text{low}\, S_{K_n}, \text{up}\, S_{K_n})$ relates to the subdivision obtained when the symbol S_{K_n} can be decoded without ambiguity, K_n is a counter representing the number of symbols that has been completely decoded at the bit instant n. Both intervals must be scaled appropriately in order to avoid numerical precision problems.

Note also that, in practical applications, the sources to be encoded are Q-ary sources. The use of a quasiarithmetic coder, if one desires to keep high compression efficiency properties as well as a tractable computational complexity, requires to first convert the Q-ary source into a binary source. This conversion amounts to consider a fixed-length binary representation of the source, as already performed in the EBCOT [58] or CABAC [59] algorithms used in the JPEG-2000 [60] and H.264 [61] standards, respectively. The full exploitation of all dependencies in the stream then requires to consider an automaton that is the product of the automaton corresponding to the source conversion and to the QA-coder/decoder automaton [30].

3.4. MAP estimation or finite-state trellis decoding

When the coder can be modelled as a finite-state automaton, MAP, MPM, or MMSE estimation of the sequence of hidden states X_0^N can be performed on the trellis representation of the automaton, using, for example, BCJR [19] and SOVA [18] algorithms. We consider as an example the product model described in Section 3.2 (see Figure 2d), with $X_n = (\nu, s)$. The symbol-by-symbol MAP estimation using the BCJR algorithm will search for the best estimate of each state X_n by computing the a posteriori probabilities (APPs) $\mathbb{P}(X_n | Y_1^N)$. The computation of the APP $\mathbb{P}(X_n | Y_1^N)$ is organized around the factorization

$$\mathbb{P}(X_n | Y_1^N) \propto \mathbb{P}(X_n, Y_1^n) \cdot \mathbb{P}(Y_{n+1}^N | X_n). \tag{6}$$

Assuming the length N of the bit sequence to be known, and the length K of the sequence of symbols to be unknown, the Markov property of the chain X_0^N allows a recursive computation of both terms of the right-hand side. A forward recursion computes

$$\begin{aligned} \alpha_n &= \mathbb{P}(X_n, Y_1^n) \\ &= \sum_{x_{n-1}} \mathbb{P}(X_{n-1} = x_{n-1}, Y_1^{n-1}) \\ &\quad \cdot \mathbb{P}(Y_n \mid X_{n-1} = x_{n-1}, X_n) \\ &\quad \cdot \mathbb{P}(X_n \mid X_{n-1} = x_{n-1}). \end{aligned} \tag{7}$$

The summation on x_{n-1} denotes all the possible realizations that can be taken by the random variable X_{n-1} denoting the state at instant $n-1$ of the FSA considered. The quantity Y_n is a measurement on the bit U_n corresponding to the transition (X_{n-1}, X_n) on the FSA. The backward recursion computes

$$\begin{aligned} \beta_n &= \mathbb{P}(Y_{n+1}^N | X_n) \\ &\propto \sum_{x_{n+1}} \mathbb{P}(X_{n+1} = x_{n+1} | X_n) \\ &\quad \cdot \mathbb{P}(Y_{n+2}^N \mid X_{n+1} = x_{n+1}) \\ &\quad \cdot \mathbb{P}(Y_{n+1} \mid X_n, X_{n+1} = x_{n+1}), \end{aligned} \tag{8}$$

where $\mathbb{P}(X_{n+1} = x_{n+1} | X_n)$ and $\mathbb{P}(Y_{n+1} \mid X_n, X_{n+1} = x_{n+1})$ denote the transition probability on the source coder automaton and the channel transition probability, respectively. The posterior marginal on each emitted bit U_n can in turn be obtained from the posterior marginal $\mathbb{P}(X_n, X_{n+1} | Y)$ on transitions of X. Variants of the above algorithm exist: for example, the log-MAP procedure performs the computation in the log domain of the probabilities, the overall metric being formed as sums rather than products of independent components.

Similarly, the sequence MAP estimation based on the modified SOVA [62, 63, 64] proceeds as a bidirectional recursive method with forward and backward recursions in order to select the path with the maximum metric. For each state, the metric corresponds to the maximum metric over

all paths up to that state, with the branch metric defined as the log-likelihood function

$$\begin{aligned} M_n(X_{n-1} = x_{n-1},\, X_n = x_n) &= \ln\left[\mathbb{P}(Y_n \mid X_{n-1} = x_{n-1},\, X_n = x_n)\right] \\ &\quad + \ln\left[\mathbb{P}(X_n = x_n \mid X_{n-1} = x_{n-1})\right]. \end{aligned} \tag{9}$$

For two states (x_{n-1}, x_n) for which a branch transition does not exist, the metric is negative infinity. The soft output on the transition symbol is obtained by combining the forward metric at instant $n-1$, the backward metric at instant n, and the metrics for branches connecting the two sets. This soft output is either expressed as the likelihood ratio, that is, as the APP ratio of a symbol to the sum of APPs of all the other symbols or as a log-likelihood ratio. The algorithm producing a log-likelihood ratio as soft output is equivalent to a Max-Log-MAP algorithm [65], where the logarithm of the exponentials of the branch metrics is approximated by the Max. Note that MMSE estimators can also be applied provided that the bit-level APPs are converted into symbol-level APP or by directly considering a symbol-level trellis representation of the source coder. For the bit-symbol conversion of APP, one can rely on the symbol counter l inside the X_n state vector to isolate states that are involved in a given symbol.

3.5. Soft-input soft-output sequential decoding

Some variable-length source coding processes, for example, optimal arithmetic coding, cannot be modelled as automata with a realistic state-space dimension. Indeed, the number of states grows exponentially with the number of symbols being encoded. In addition, in the case of arithmetic coding, the state value is dependent on all the previous states. In this case, sequential decoding techniques such as the Fano algorithm [66] and the stack algorithm [67], initially introduced for convolutional codes, can be applied. Sequential decoding has been introduced as a method of ML sequence estimation with typically lower computation requirements than those of the Viterbi decoding algorithm, hence allowing for codes with large constraint lengths. The decoding algorithm follows directly the coder/decoder decision tree structure. Any set of connected branches through the tree, starting from the root, is termed a path. The decoder examines the tree, moving forward and backward along a given path according to variations of a given metric. The Fano algorithm and metric, initially introduced for decoding channel codes of both fixed and variable length [68], without and with [69] a priori information, is used in [70] for error-resilient decoding of MPEG-4 header information, in [71] for sequential soft decoding of Huffman codes, and in [72] for JSCD.

Sequential decoding has been applied to the decoding of arithmetic codes in [23], assuming the source to be white. A priori source information can in addition be exploited by modifying the metric on the branches. A MAP metric, similar to the Fano metric, can be considered and defined as the APP of each branch of the tree, given the corresponding set of observations, leading to sequential decoding with soft output. This principle has been applied in [24, 25] for error-resilient decoding of arithmetic codes, with two ways of formalizing the MAP metric. Given that S_1^K uniquely determines U_1^N and vice-versa, the problem of estimating the sequence S_1^K given the observations Y_1^N can be written as [24]

$$\begin{aligned} \mathbb{P}(S_1^K \mid Y_1^N) &= \mathbb{P}(U_1^N \mid Y_1^N) \\ &\propto \mathbb{P}(S_1^K) \cdot \mathbb{P}(Y_1^N \mid S_1^K) \\ &= \mathbb{P}(S_1^K) \cdot \mathbb{P}(Y_1^N \mid U_1^N). \end{aligned} \tag{10}$$

The quantity $\mathbb{P}(S_1^K \mid Y_1^N)$ can be computed recursively as

$$\begin{aligned} \mathbb{P}(S_1^k \mid Y_1^{N_k}) &\propto \mathbb{P}(S_1^{k-1} \mid Y_1^{N_{k-1}}) \\ &\quad \cdot \mathbb{P}(S_k \mid S_{k-1}) \cdot \mathbb{P}(Y_{N_{k-1}+1}^{N_k} \mid Y_1^{N_{k-1}}, U_1^{N_k}), \end{aligned} \tag{11}$$

where N_k is the number of bits that have been transmitted when arriving at the state X_k. Assuming S_1^K to be a first-order Markov chain and considering a memoryless channel, this APP can be rewritten as

$$\begin{aligned} \mathbb{P}(S_1^k \mid Y_1^{N_k}) &\propto \mathbb{P}(S_1^{k-1} \mid Y_1^{N_{k-1}}) \\ &\quad \cdot \mathbb{P}(S_k \mid S_{k-1}) \cdot \mathbb{P}(Y_{N_{k-1}+1}^{N_k} \mid U_{N_{k-1}+1}^{N_k}). \end{aligned} \tag{12}$$

Different strategies for scanning the branches and searching for the optimal branch of the tree can be considered. In [23], the authors consider a depth-first tree searching approach close to a Fano decoder [66] and a breadth-first strategy close to the M-algorithm, retaining the best M paths at each instant in order to decrease the complexity. In [25], the authors consider the *stack algorithm* (SA) [73].

Figure 3 illustrates the symbol error rate (SER) performance obtained with a first-order Gauss-Markov source with zero-mean, unit variance, and correlation factors $\rho = 0.9$ and $\rho = 0.5$. The source is quantized on 8 levels. The channel is an AWGN channel with a signal-to-noise ratio varying from $E_b/N_0 = 0$ dB to $E_b/N_0 = 6$ dB. Figure 3 shows a significant SER performance gap between Huffman and arithmetic codes when using decoding with soft information. The performance gap between Huffman codes and arithmetic codes decreases with decreasing correlation, however remains at the advantage of arithmetic codes. The gain in compression performance of arithmetic codes gives extra freedom to add some controlled redundancy, for example, in the form of soft synchronization patterns (see Section 4), that can be dedicated to fight against the desynchronization problem. This problem indeed turns out to be the most crucial problem in decoding VLC-coded sources.

The sequential decoding algorithm presented above has been used in an iterative structure following the turbo principle in [24] for JSCD of arithmetic codes. The APP $\mathbb{P}(S_1^K, N_K = N \mid Y_1^N)$ (the quantity $N_K = N$ meaning that only the paths corresponding to the number of bits received are kept) on the last state corresponding to entire sequences of symbols are thus converted into APP on bits U_n by the

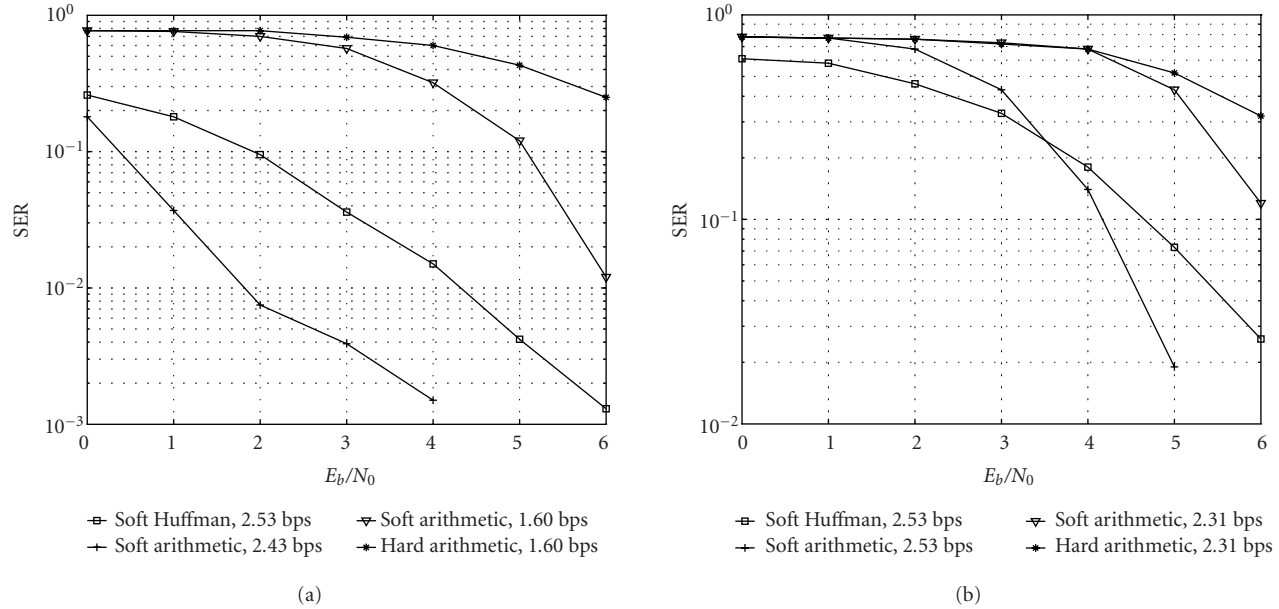


Figure 3: SER performances of soft arithmetic decoding, hard arithmetic decoding, and soft Huffman decoding (for (a) $\rho = 0.9$ and (b) $\rho = 0.5$, 200 symbols, 100 channel realizations, courtesy of [24]).

equation

$$\widetilde{P}(U_n = i|Y)|_{i=0,1} \propto \sum_{\text{all surviving paths } s_1^K : U_n = i} \mathbb{P}(s_1^K, N_K|Y), \quad (13)$$

where $\propto$ denotes an obvious renormalization. The tilde in the term $\widetilde{P}$ denotes the fact that these probability values are only approximations of the real APP on the bits U_n, since only the surviving paths are considered in their computation. However, the gain brought by the iterations is small. This is explained both by the pruning needed to maintain the decoding complexity within a realistic range, and by the fact that the information returned to the channel decoder is approximated by keeping only the surviving paths.

Remark 1. Quasiarithmetic coding can be regarded as a reduced precision implementation of arithmetic coding. Reducing the precision of the coding process amounts to approximate the source distribution, hence in a way to leave some redundancy in the compressed stream. A key advantage of quasiarithmetic versus arithmetic codes comes from the fact that the coding/decoding processes can be modelled as FSA. Thus, efficient trellis decoding techniques, such as the BCJR algorithm, with tractable complexity can be used. In presence of transmission errors, QA codes turn out to outperform arithmetic codes for sources with low to medium ($\rho \leq 0.5$) correlation. However, for highly correlated sources, the gain in compression brought by optimal arithmetic coding can be efficiently exploited by inserting, up to a comparable overall rate, redundancy dedicated to fight against the critical desynchronization problem, leading to higher SER and SNR performance.

3.6. A variant of soft-input soft-output VLC source decoding with factored models

To reduce the state-space dimension of the model or trellis on which the estimation is run, one can consider separate models for the Markov source and the source coder. It is shown in [14] that a soft source decoding followed by a symbol stream estimation is an optimal strategy. Notice that this is possible only if the model of dependencies (hence the automaton) of the decoder is not a function of previous source symbol realizations. For example, we consider a Huffman coder with a unique tree constructed according to stationary probabilities. As explained above, to take into account the intersymbol correlation, one changes the transition probabilities on this unique tree according to the previous symbol realization (for first-order Markov sources), however, the automaton structure remains the same. One can hence consider separately the automaton corresponding to the codetree structure and the automaton corresponding to the Markov source. The resulting network of dependencies following a tree-structure, the Markov source, and the source coder need not be separated by another interleaver to design an optimum estimator.

To separate the two models, one must however be able to translate pointwise measurements Y_1^N on the useful bits U_1^N into measurements on symbols. This translation is then handled via the two augmented Markov processes: (S, N) composed of pairs (S_k, N_k) which represents the Markov source and (X, K) composed of pairs (X_n, K_n) representing the coding process described in Section 3 [14]. The estimation can actually be run in two steps as follows.

(i) The first step consists in estimating states (X_n, K_n) assuming symbols are independent, which uses only the inner-codeword redundancy and the constraint

on K; this amounts to computing the probabilities $P(X_n, K_n|Y)$, which can be done by a standard BCJR algorithm.

(ii) The symbol stream is in turn estimated using the symbol-clock HMM to exploit the intersymbol correlation. This second step, being performed on the symbol clock model of the source, requires as inputs the posterior marginals $\mathbb{P}(S_k, N_k|\overline{Y}_k)$, hence requires a translation of the distributions $P(X_n, K_n|Y)$ into symbol-level posterior marginals $\mathbb{P}(S_k, N_k|\overline{Y}_k)$, where $\overline{Y}_k$ represents the variable length set of measurements on the codeword $\overline{U}_k$ associated to the symbol S_k. This conversion is made possible with the presence of the counters N_k and K_n.

Now, we assume that an optimal Huffman coder is considered for the first-order Markov source. This requires to use multiple codetrees according to the last symbol realization, and this in order to follow the source conditional probability. In that case, the structure of the decoder automaton changes at each symbol instant, impeding the separation of the two models. This is the case of quasiarithmetic and arithmetic coders and decoders. The corresponding coding processes indeed follow the conditional distribution of the source. Hence, at a given symbol instant, the decoding automaton is dependent on the last decoded symbol realization. This is also the case for optimal arithmetic coding and decoding for which a state (defined by the bounds of probability intervals) depends on all the previous symbol realizations.

4. SYNCHRONIZATION AND ERROR DETECTION IN SOFT DECODING OF VLCs

We have seen in Section 3 that if the number of symbols and/or bits transmitted are known by the decoder, termination constraints can be incorporated in the decoding process: for example, one can ensure that the decoder produces the right number of symbols ($K_N = K$) (if known). All the paths in the trellis which do not lead to a valid sequence length are suppressed. The termination constraints mentioned above allow to synchronize the decoding at both ends of the sequence but however do not guarantee synchronous decoding of the middle of the sequence. Extra synchronization and error detection mechanisms can be added as follows.

(i) *Soft synchronization.* One can incorporate extra bits at some known positions $I_s = \{i_1, \ldots, i_s\}$ in the symbol stream to precisely help achieving a proper segmentation of the received noisy bitstream into segments that will correspond to the symbols that have been encoded. This extra information can take the form of dummy symbols (in the spirit of the techniques described in [23, 26, 27, 29, 74]), or of dummy bit patterns which are inserted in the symbol or bitstream, respectively, at some known symbol clock positions. Bit patterns can have arbitrary length and frequency, depending on the degree of redundancy desired. The procedure amounts to extending symbols at known positions with a suffix $\overline{U}_{K_n} \Rightarrow \overline{U}_{K_n} B_1 \cdots B_{l_s}$, of a given length l_s. Transitions are deterministic in this extra part of the tree. These suffixes favor the likelihood of correctly synchronized sequences (i.e., paths in the trellis), and penalize the others.

(ii) *Error detection and correction based on a forbidden symbol.* To detect and prune erroneous paths in soft arithmetic decoding, the authors in [23, 25] use a reserved interval corresponding to a so-called *forbidden symbol*. All paths hitting this interval are considered erroneous and pruned.

(iii) *Error detection and correction based on a CRC.* The suffixes described for soft synchronization can also take the form of a cyclic redundancy check (CRC) code. The CRC code will then allow to detect an error in the sequence, hence pruning the corresponding erroneous path.

The termination constraints do not induce any redundancy (if the numbers of bits and symbols transmitted are known; otherwise, the missing information has to be transmitted) and can be used by any VLC soft decoder to resynchronize at both ends of the sequence, whatever the channel characteristics. The other approaches, that is, soft synchronization, forbidden symbol, or CRC help the decoder to resynchronize at intermediate points in the sequence, at the expense of controlled redundancy. A complete investigation of the respective advantages and drawbacks of the different techniques for different VLCs (e.g., Huffman, arithmetic codes) and channel characteristics (e.g., random versus bursty errors, low versus high channel SNR) is still to be carried out.

5. JOINT SOURCE-CHANNEL DECODING WITH SOFT INFORMATION

In this section, we consider the case where there is a recursive systematic convolutional (RSC) coder in the transmission chain. The channel coder produces the redundant bitstream R by filtering useful bits U according to

$$R(z) = \frac{F(z)}{G(z)} U(z), \tag{14}$$

where $F(z)$ and $G(z)$ are binary polynomials of maximal degree δ, z denoting the delay operator. Once again, this filtering can be put into state-space form by taking the RSC memory content m as a state vector. This makes the coder state a Markov chain, with states denoted $X'_n = m$, when the coder is driven by a white noise sequence of input bits. Optimal decoding requires to make use of both forms of redundancy, that is, of the redundancy introduced by the channel code and of the redundancy present in the source-coded bitstream. This requires to provide a model of the dependencies present in the complete source-channel coding chain.

5.1. Product model of dependencies

To get an exact model of dependencies amenable to optimal estimation, one can build a product of the three models (source, source coder, channel coder) with state vectors $X_k = (\nu, s, l, m)$ in the case of the codetree-based coder, where ν, s, l are state variables of the source and source coder models, as defined in Section 3. In the case of a QA coder, the state vectors would be $X_k = ([\mathrm{low}_k, \mathrm{up}_k), m)$. Such a product

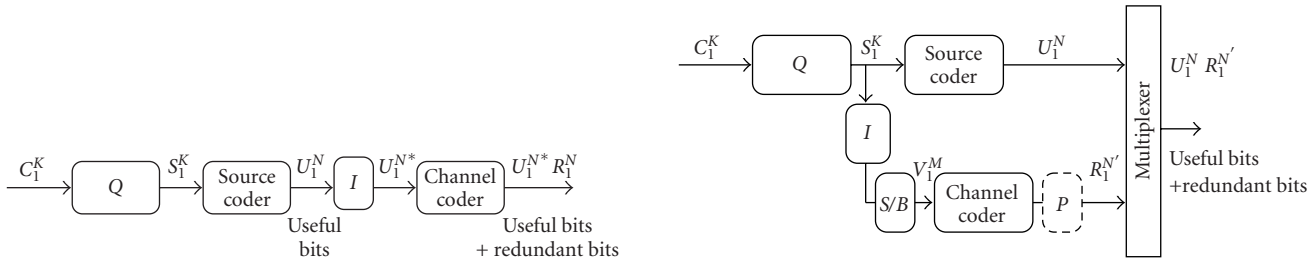


FIGURE 4: (a) Serial and (b) parallel joint source-channel coding structures. I denotes an interleaver, P an optional puncturing mechanism, and S/B a symbol-to-bit conversion. The example depicted in the serial structure assumes a systematic channel coder of rate 1/2. In the parallel structure, V_1^M denotes the binary representation of the quantized source symbol indexes. To have an overall rate equivalent to the one given by the serial structure, the code rate and puncturing matrix can be chosen so that $N' = N$.

model gathering state representations of the three elements of the chain has been proposed in [16]. The set of nodes is thus the product of the nodes in the constituent graphs, each node of the joint decoder graph containing state information about the source, the source code, and the channel code. The resulting automaton can then be used to perform a MAP, MPM, or MMSE decoding. The approach allows for optimal joint decoding, however, its complexity remains untractable for realistic applications. The state-space dimension of the product model *explodes* in most practical cases, so that a direct application of usual techniques is unaffordable, except in trivial cases. Instead of building the Markov chain of the product model, one can consider the serial or parallel connection of two HMMs, one for the source + source coder (or separately for the source and source coder as described above) and one for the channel coder, in the spirit of serial and parallel turbo codes. The dimension of the state space for each model is then reduced.

The direct connection of the two HMMs (the source coder HMM and the channel coder HMM) would result in a complex dependency (Bayesian) network with a high number of short cycles, which is, as such, not amenable to fast estimation algorithms. However, it has been observed with turbo codes [75, 76, 77] that efficient approximate estimation could be obtained by proceeding with the probabilistic inference in an iterative way, making use of part of the global model at each time, provided the cycles in the network of dependencies are long enough. It was also observed that the simple introduction of an interleaver between two models can make short cycles become long. The adoption of this principle, known as the *turbo* principle [78], led to the design of iterative estimators working alternatively on each factor of the product model. The estimation performance obtained is close to the optimal performance given by the product model.

5.2. Serially concatenated joint source-channel (de-) coding

This principle has been applied to the problem of joint source-channel decoding by first considering a serial concatenation of a source and a channel coder, as shown in Figure 4a. Figure 5 shows the structure of the corresponding iterative decoder. In Figure 5, it is assumed that the channel encoder is a systematic convolutional code with rate 1/2 that provides systematic bits, denoted by U_1^N, and redundant bits R_1^N. However, the principle applies similarly to channel codes of different rates. Based on the schematic representations given in Figure 4a, it has to be noted that U_1^N denotes an interleaved sequence. After transmission through a noisy channel, the decoder receives the corresponding observations, denoted by Y_1^N and Z_1^N, respectively. In Figure 5, the channel and source decoders are composed of soft-input soft-output (SISO) decoding components.[3] The SISO components for the source decoder can be either a trellis decoder (e.g., using BCJR or SOVA algorithms) or a sequential decoder, as described in Section 3.5. The two decoding components are separated by an interleaver I and a deinterleaver I^*.[4]

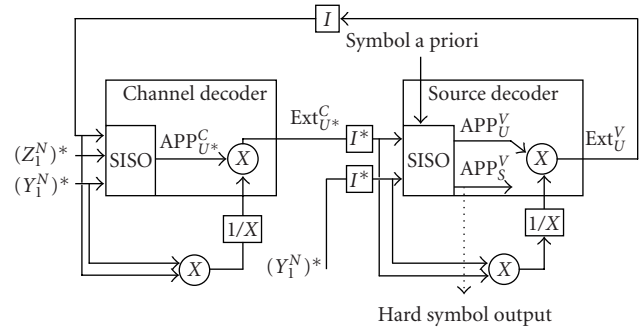


FIGURE 5: Joint source-channel decoding structure for a serial source-channel encoder (courtesy of [79]).

[3] Note that additional bits can be used for terminating the trellis, but this is not absolutely necessary. For instance, the results reported in Figure 9 are obtained considering uniform probabilities for initializing the different states of the RSC encoder in the BCJR backward recursion.

[4] The notation $*$ is used to represent an interleaved sequence.

A first estimation (BCJR or SOVA) is run on the channel decoder HMM with measures $Y_1^{N^*}$ on the interleaved sequence of useful bits and the sequence of measures Z_1^N on the redundant bits as inputs. It involves the computation of a sequence of APPs for the interleaved sequence $U_1^{N^*}$ denoted by $\mathrm{APP}^C_{U^*}$. Then, the extrinsic information $\mathrm{Ext}^C_{U_n^*}$ relative to each bit U_n^* of the interleaved sequence $U_1^{N^*}$ of useful bits is computed from its posterior distribution obtained as a result of the channel decoding. The *extrinsic* information can be regarded as the modification induced by a new measurement (here all the measures $Y_1^* \cdots Y_N^*$ except for the *local* one Y_n^*) on the APPs on the interleaved useful bits U_n^* conditioned by the *local* measurement Y_n^*. It can also be regarded as the incremental information on a current decoder state through the estimation of all the other decoder states. This extrinsic information is computed as

$$\mathrm{Ext}^C_{U_n^*}\left(Y^* = y^* \mid Y_n^* = y_n^*\right) = \frac{\mathbb{P}\left(U_n^* \mid Y^* = y^*\right)}{\mathbb{P}\left(U_n^* \mid Y_n^* = y_n^*\right)\mathrm{Ext}^V_{U_n^*}\left(Y^* = y^* \mid Y_n^* = y_n^*\right)}, \tag{15}$$

where $\mathrm{Ext}^V_{U^*}$ represents the interleaved sequence of the extrinsic information produced by the VLC decoder. Note that, when running the first channel SISO decoder (i.e., at iteration 0), this term simplifies as

$$\mathrm{Ext}^C_{U_n^*}\left(Y^* = y^* \mid Y_n^* = y_n^*\right) = \frac{\mathbb{P}\left(U_n^* \mid Y^* = y^*\right)}{\mathbb{P}\left(U_n^* \mid Y_n^* = y_n^*\right)}. \tag{16}$$

If the estimation is run in a logarithmic domain, the extrinsic information is computed by subtracting the logarithm of the probability laws. The *extrinsic* information on a useful bit is a direct subproduct of a BCJR algorithm or of a SOVA. In the case of sequential decoding, a conversion of the APP on the entire sequence of symbols (or equivalently states of the decoder) into the APP of each useful bit, as expressed in (13), is needed. Notice that the motivation for feeding only extrinsic information from one stage to the next is to maintain as much statistical independence between the bits as possible from one iteration to the next. As long as iterative decoding proceeds, and assuming sufficient interleaving, the reliability on states (or on transition bits) improves until it gets to a constant value. If the assumption of statistical independence is true, the iterative estimation on parts of the model each time approaches the MAP solution on the global model of dependencies as the number of iterations approaches infinity.

Thus, the channel decoder produces a sequence of extrinsic information ($\mathrm{Ext}^C_{U^*} = \mathrm{Ext}^C_{U_1^*} \cdots \mathrm{Ext}^C_{U_N^*}$) which is deinterleaved before being fed into the VLC decoder. A similar computation has to be carried out in the source decoder considering the deinterleaved versions Y_1^N and Ext^C_U of the sequences of measurements and extrinsic information. It, in turn, involves a computation of a sequence of APPs (APP^V_U) and yields another sequence of extrinsic information on the useful bits:

$$\mathrm{Ext}^V_{U_n}\left(Y = y \mid Y_n = y_n\right) = \frac{\mathbb{P}\left(U_n \mid Y = y\right)}{\mathbb{P}\left(U_n \mid Y_n = y_n\right)\mathrm{Ext}^C_{U_n}\left(Y = y \mid Y_n = y_n\right)}. \tag{17}$$

The sequence Ext^V_U of extrinsic information is interleaved and then fed into the channel decoder. After a few iterations involving the two SISO decoders, the source decoder outputs the symbol estimates.

This principle has been very largely applied to joint source-channel coding and decoding of fixed-length [32] and variable-length (Huffman, RVLC, arithmetic, quasiarithmetic) codes. The convergence behavior of iterative source-channel decoding with fixed-length source codes and a serial structure is studied in [33] using extrinsic information transfer (EXIT) charts [34]. The gain brought by the iterations is obviously very much dependent on the amount of correlation present on both sides of the interleaver. The variants of the algorithms proposed for joint source-channel decoding of VLC-encoded sources relate to various forms of trellis representations for the source coder, as seen in Section 3, as well as to the different underlying assumptions with respect to the knowledge of the length of the sequences of symbols or of bits [12, 13, 14, 17, 20, 21, 35, 52].

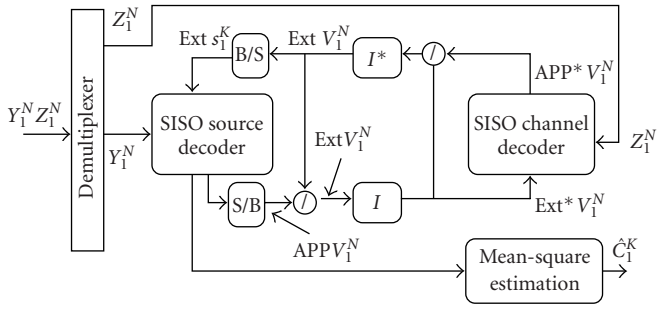


Figure 6: Parallel iterative joint source-channel decoding structure.

5.3. Parallel-concatenated joint source-channel decoding

A parallel-concatenated source-channel coding and decoding structure with VLC-encoded sources is described in [38]. In comparison with a parallel channel turbo coder, the explicit redundancy from one channel coder is replaced by redundancy left in the source compressed stream U_1^N (see Figure 4b) after VLC encoding. The indexes of the quantized symbols are converted into a sequence of bits V_1^M which is fed into a channel coder (possibly followed by a puncturing matrix to adjust the channel code rate). The channel coder produces the sequence of parity bits $R_1^{N'}$. The decoder (see Figure 6) proceeds with an iterative estimation where the source decoder computes first the APPs on the quantized symbol indexes, $\mathrm{APP}(S_k)$, which are then converted into APPs on the bit representation of the indexes ($\mathrm{APP}(V_1^M)$).

Extrinsic information on the binary representation of the quantized indexes, $\text{Ext}(V_1^M)$, is then computed by removing (via a subtraction or a division depending on whether the estimation is run in a logarithmic domain or not) the a priori information. The interleaved extrinsic information, $\text{Ext}^*(V_1^M)$, is fed as a priori information to the soft-input soft-output channel decoder. Extrinsic information resulting from the estimation run on the channel decoder model, after deinterleaving, is converted into a priori information on quantized symbols, which is fed in a second iteration to the soft-input soft-output source decoder. The authors in [38] show that, after the 20th iteration and for almost the same code rate (around 0.3), the parallel structure brings a gain that may be up to 3 dB in terms of SNR of the reconstructed source signal with respect to the serial structure. However, this result, that is, the superiority of the parallel versus serial structure, analogous to the comparison made between parallel and serial turbo codes [80], is limited to the case of a given RVLC code and to a AWGN channel with low SNRs.

6. SOURCE-CONTROLLED CHANNEL DECODING

Another possible approach is to modify the channel decoder in order to take into account the source statistics and the model associated to the source and source coder. A key idea presented in [39] is to introduce a slight modification of a standard channel decoding technique in order to take advantage of the source statistics. This idea has been explored at first in the case of FLC and validated using convolutional codes in a context of transmission of coded speech frames over the global system of mobile telecommunication (GSM). Source-controlled channel decoding have also been applied with block and convolutional turbo codes, considering FLC for hidden Markov sources [40] or images [41, 81, 82, 83]. The authors in [81], by first optimizing the turbo code polynomials, and second by taking into account source a priori information in the channel decoder, show performances closer to the optimal performance theoretically achievable (OPTA) in comparison with a tandem decoding system based on Berrou's rate-1/3 $(37, 21)$ turbo code, for the same overall rate. However, when using FLC source codes, the excess rate in the bit sequence fed into the channel coder is high. The source has not been compressed, and the channel code rate is high. To draw any conclusion on the respective advantages of joint versus tandem source-channel decoding techniques, one must consider the chain in which the source has been compressed as well. The freed bandwidth may then allow to reduce the channel code rate, hence increasing the error correction capability of the channel code. In this section, we show how the approach of source-controlled channel decoding can be extended to cover the case of JSCD with VLC.

6.1. Source-controlled convolutional decoding with VLCs

Source-controlled channel decoding of VLC-coded sources has been first introduced in [42]. The transmission chain considered is depicted in Figure 1: the source compressed stream produced by a VLC coder is protected by a convolutional code. The convolutional decoder proceeds by running a Viterbi algorithm which estimates the ML sequence. If we denote by X_0^N the sequence of states of the convolutional encoder, the ML estimation searches for the sequence X_0^N such that $\mathbb{P}(Y_1^N|X_0^N)$ is maximum. The ML estimate would be equivalent to the MAP estimate if the source was equiprobably distributed, that is, if the quantity $\mathbb{P}(X_n|X_{n-1})$ is constant.

However, here, the input U of the channel coder is not in general a white sequence but a pointwise function of a Markov chain. The quantity $\mathbb{P}(X_n|X_{n-1})$ is therefore not any more constant, but has instead to be derived from the source statistics. One has in this case to use instead the generalized Viterbi algorithm [44] in order to get the optimal MAP sequence, that is, the one minimizing the number of bit errors. For this, a one-to-one correspondence has to be maintained between each stage of the decoding path in the convolutional decoder and the vertex in the VLC tree [42] associated to the corresponding useful bit U_n at the input of the channel coder. The probability $\mathbb{P}(X_n|X_{n-1})$ is thus given by the transition probability on the VLC codetree. For a first-order Markov source, to capture the intersymbol correlation, the probability $\mathbb{P}(X_n|X_{n-1})$ becomes dependent on the last symbol that has been coded, as explained in Section 3. The decoding algorithm thus proceeds with the search for the path that will maximize the metric

$$\max_{\chi} \mathbb{P}(X_0^N|Y_1^N) \Longleftrightarrow \max_{\chi} \left(\sum_{n=1}^{N} \ln \mathbb{P}(Y_n|X_n, X_{n-1}) + \ln \prod_{n=1}^{N} \mathbb{P}(X_n|X_{n-1}) \right), \tag{18}$$

where χ denotes the set of all possible sequences of states for the channel decoding trellis [43, 79]. Results reported in [42, 84] show that though this method is suboptimal, it nevertheless leads to performances that are close to the ones provided by the optimum MAP decoder [16], for which a product of the Markov source model, of the source coder, and channel coder model is computed.

6.2. Source-controlled turbo decoding with VLCs

Source-controlled turbo decoding can also be implemented for VLC compressed sources. In the transmission system considered in [42, 84], the symbol stream $S_1, S_2, \ldots, S_K$ is encoded using a VLC followed by a systematic turbo code which is a parallel concatenation of two convolutional codes. The transmitted stream, denoted by $U_1, U_2, \ldots, U_N, R_1, R_2, \ldots, R_N$ in Figure 1, now corresponds to a sequence of N triplets, denoted by $(U_n, R_{n,1}, R_{n,2})$, where U_n denotes the systematic bits and $R_{n,1}$, $R_{n,2}$ the parity bits from the two constituent encoders. In contrast to Section 5, U_1^N now designates a sequence of noninterleaved bits. In order to decode according to the "turbo principle," an extrinsic information has to be computed for each information bit. To achieve this task, several algorithms can be used [39, 48, 75].

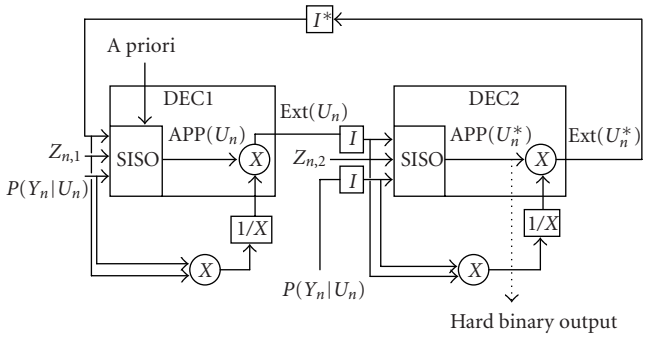


Figure 7: Parallel turbo decoding structure using a priori source information in the first decoder (courtesy of [79]).

Assuming that each decoder, say DEC1 and DEC2, can be represented by an M-state trellis, then each decoder computes the APP of each information bit as

$$\mathrm{APP}\,(U_n) = \mathbb{P}(U_n|Y_1^N) = \sum_{X_n=1}^{M} \mathbb{P}(U_n, X_n|Y_1^N). \qquad (19)$$

As shown in [75], the explicit expression of $\mathrm{APP}(U_n)$ involves a term corresponding to the state transition probability $\mathbb{P}(X_n|X_{n-1})$, given, as in the case of source-controlled convolutional code decoding, by the source statistics. The source information is actually exploited only in the first decoder. The procedure is illustrated in Figure 7 in the case of a parallel turbo encoder where the triplet $(Y_n, Z_{n,1}, Z_{n,2})$ corresponds to the systematic bit and the two parity bits, respectively. To reduce the complexity, a submap algorithm can be used in the first decoder (DEC1) [42, 79].

Figure 8 shows the SER and the Levenshtein distance curves obtained with a tandem decoding system not taking into account a priori source information and with a JSCD scheme, where the first constituent decoder takes advantage of this a priori information. The source considered is a very simple 3-symbol first-order Gauss-Markov source compressed with a Huffman code governed by the source stationary distribution [16, 53]. The turbo encoder is composed of two RSC codes defined by the polynomials $F(z) = 1 + z + z^2 + z^4$ and $G(z) = 1 + z^3 + z^4$. The parity bits are punctured in order to get a code rate equal to 1/2. A 64×64 line-column interleaver is inserted between the two constituent codes. The simulations have been carried out over an AWGN channel characterized by its signal-to-noise ratio, E_b/N_0, with E_b the energy per useful transmitted bit and N_0 the single-sided noise density. For two different measures of the SER, a standard one based on the standard direct computation and a second one using the Levenshtein distance [85], it is shown, for the first three turbo decoding iterations, that the JSCD scheme provides a significant improvement compared to the tandem scheme. Furthermore, this high gain, that can reach 2.1 dB, can be obtained for a large range of SER values (whatever the measure being used). Note that, in this scheme, the decoding is based on a Max-Log-MAP algorithm. However, one could alternatively use a modified version of the SOVA algorithm described in [62].

Source-controlled turbo decoding has also been studied with RVLC in [86] where the authors show that higher performance could be achieved in comparison with Huffman codes.

7. ESTIMATION OF SOURCE STATISTICS FROM NOISY OBSERVATIONS

In a practical set-up, in order to run the above algorithms, the source statistics need to be estimated from the received noisy bitstream Y_1^N. If we consider a quantized first-order Markov source, both the stationary and conditional distributions ($\mathbb{P}(S_k)$ and $\mathbb{P}(S_k|S_{k-1})$) need to be estimated. Two cases can be considered: if the source can be represented using a reasonable number of parameters, a parametric estimation can be carried out, otherwise, a direct estimation has to be performed.

In [82] where the VQ indexes are coded with FLC, a direct estimation using simple histograms of stationary and transition probabilities is shown to be sufficient. However, this assumes the source to be stationary. Alternatively, a parametric estimation method, making use of a modified turbo decoding procedure, is described in [88]. The estimation procedure has been tested with a quantized first-order Gauss-Markov (GM) source having a correlation factor denoted by ρ_S. It is shown that for a stationary source, an appropriate solution is to proceed, before iterating, to a hard decoding at the source decoder output. Then the correlation, say ρ, can be estimated using a Yule-Walker algorithm. From this correlation, we can easily get an estimate of the transition probabilities that are used at the next iteration to help the BCJR decoding of the source. Setting the initial value of ρ to zero, it is shown that, for sufficiently high channel signal-to-noise ratio (E_b/N_0), after a few iterations, the performances obtained are close to those resulting from a perfect knowledge of ρ. Figure 9 shows a set of results obtained using a GM source, with a correlation factor $\rho_S = 0.9$, uniformly quantized on 16 levels and encoded with a Huffman code adapted to the stationary probabilities. The encoded source bitstream is protected by an RSC code (see (14)) with feed-forward and feedback polynomials given by $F(z) = 1 + z^2 + z^3 + z^4$ and $G(z) = 1 + z + z^4$, respectively. Furthermore a puncturing matrix with first and second rows given by [111], [100], respectively, is introduced that leads us to a code rate $R_c = 3/4$. After a binary phase shift keying modulation, the resulting bitstream is transmitted over an AWGN channel. In Figure 9, it can be seen that the online estimation provides acceptable results. It also appears that an overestimation setting a JSCD with $\rho = 0.9$, instead of $\rho_S = 0.5$ for the actual source, may have a dramatic impact on the overall performance.

When dealing with real nonstationary signals, a good fit with a parametrical model cannot always be found. An example is given in [89] for a model that is supposed to fit

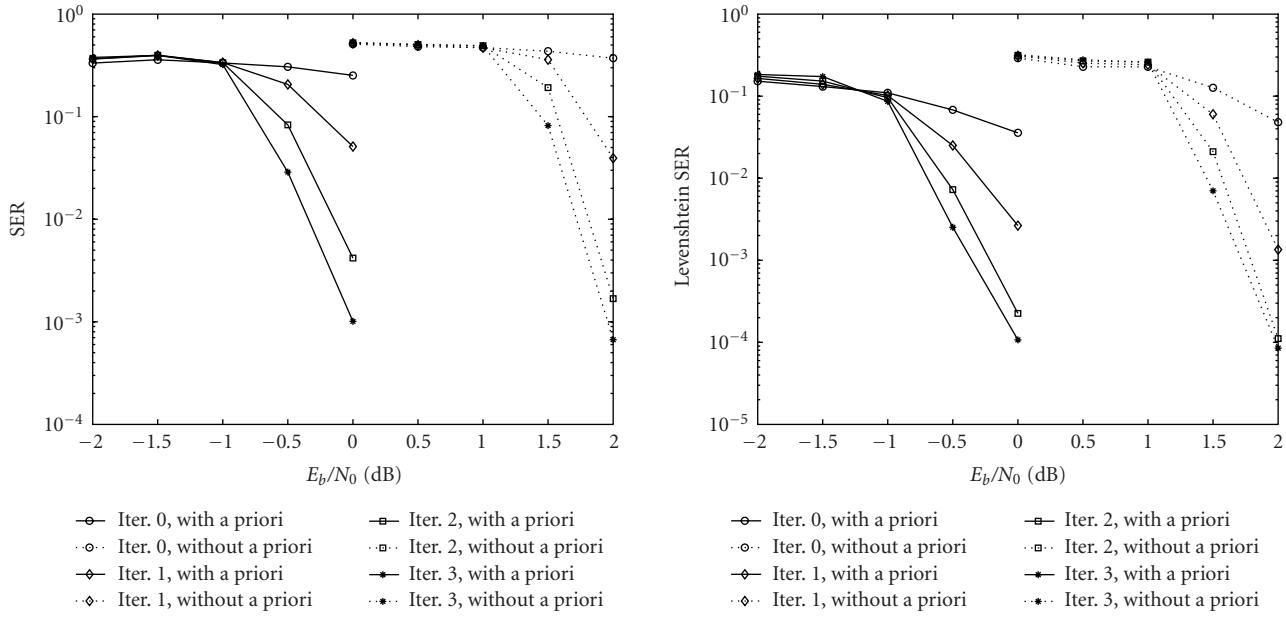


FIGURE 8: SER obtained with and without a priori information for a first-order Gauss-Markov source compressed with a Huffman code governed by the source stationary distribution (courtesy of [53]).

the motion vectors of a video sequence, where the authors acknowledge the relative inaccuracy of their model. Then a direct approach has to be preferred, the underlying problem being to estimate the parameters of an HMM. Techniques, such as the Baum-Welch algorithm [90], are well suited for this problem. They have been used in [40] for joint turbo decoding and estimation of Hidden Markov sources, and also in [12] where the authors have proposed a method based on a forward-backward recursion to estimate the HMM parameters of a VLC compressed source. In [88], an iterative source-channel decoder is slightly modified in order to integrate in its source decoding module an estimation of the source statistics. The principle is illustrated in Figure 10 where the block named SISO-VLC realizes a BCJR decoding of the VLCs using the decoding trellis presented in [14]. The SISO-VLC makes use of an augmented HMM, as explained in Section 3, in order to handle the bitstream segmentation problem. The HMM thus incorporates in the state variables a counter N_k of the number of bits encoded at the symbol instant k.

Indeed, any symbol S_k may be represented by a binary word whose length $\mathcal{L}(S_k)$ is not a priori known. Consequently, at each symbol time k ($k = 1, 2, \ldots, K$), not only the symbol S_k, but also the segmentation value $N_k = \sum_{j=1}^{k} \mathcal{L}(S_j) = N_{k-1} + \mathcal{L}(S_k)$ has to be estimated. Using the notation presented in Section 3.6, the codeword associated to S_k may be written as $\overline{U}_k = U_{N_{k-1}+1}, \ldots, U_{N_k}$.

We assume again that the symbols S_k take their values in the alphabet $\mathcal{A} = \{a_1, \ldots, a_i, \ldots, a_Q\}$. Let $\overline{Y}_{N_{k-1}}^{N_k}$ be the sequence of bits received (or of measurements) between the time instants N_{k-1} and N_k by the source decoder. The BCJR algorithm computes, for each possible realization of S_k, S_{k-1} and for each possible realization n_k of N_k,

$$\begin{aligned}
\alpha_k(a_i, n_k) &= \mathbb{P}(N_k = n_k,\ S_k = a_i,\ Y_1^{n_k}),\\
\beta_k(a_i, n_k) &= \mathbb{P}(Y_{n_k+1}^{N} \mid N_k = n_k,\ S_k = a_i),\\
\gamma_k(a_i, a_j, n_k) &= \mathbb{P}(S_k = a_i,\ N_k = n_k,\ Y_{n_k-\mathcal{L}(a_i)+1}^{n_k} \mid S_{k-1} = a_j)\\
&= \mathbb{P}(S_k = a_i \mid S_{k-1} = a_j)\\
&\quad \times \prod_{l=1}^{\mathcal{L}(a_i)} \mathbb{P}(Y_{n_k-\mathcal{L}(a_i)+l} \mid U_{n_{k-\mathcal{L}(a_i)+l}}).
\end{aligned} \tag{20}$$

Then, as in the original BCJR algorithm [19], $\alpha_k(a_i, n_k)$ and $\beta_k(a_i, n_k)$ are obtained by recursion equations corresponding to the forward and backward steps, respectively.

But in many practical problems, the source conditional probability $\mathbb{P}(a_i|a_j)$ is not a priori known and has to be estimated. The solution proposed in [87, 88] makes use of the Baum-Welch method (cf. [90] for a tutorial presentation). As the Baum-Welch source HMM parameter estimation can be carried out together with the estimation performed by the BCJR algorithm, this approach does not imply a significant increase in complexity. For a first-order Markov source and a source alphabet of size $|Q|$, the estimates of the $|Q|^2$ source conditional probabilities $\mathbb{P}(a_i|a_j)$ are estimated as

$$\widehat{\mathbb{P}}(a_i|a_j) = \frac{\sum_k \xi_k(a_i; a_j)}{\sum_k \sum_i \xi_k(a_i; a_j)},$$

$$\xi_k(a_i; a_j) = \frac{\sum_{n_k} \alpha_{k-1}(n_k - \mathcal{L}(a_i), a_j)\, \gamma_k(a_i, a_j, n_k)\, \beta_k(a_i, n_k)}{\sum_{n_k} \sum_{a_i} \sum_{a_j} \alpha_{k-1}(n_k - \mathcal{L}(a_i), a_j)\, \gamma_k(a_i, a_j, n_k)\, \beta_k(a_i, n_k)}. \tag{21}$$

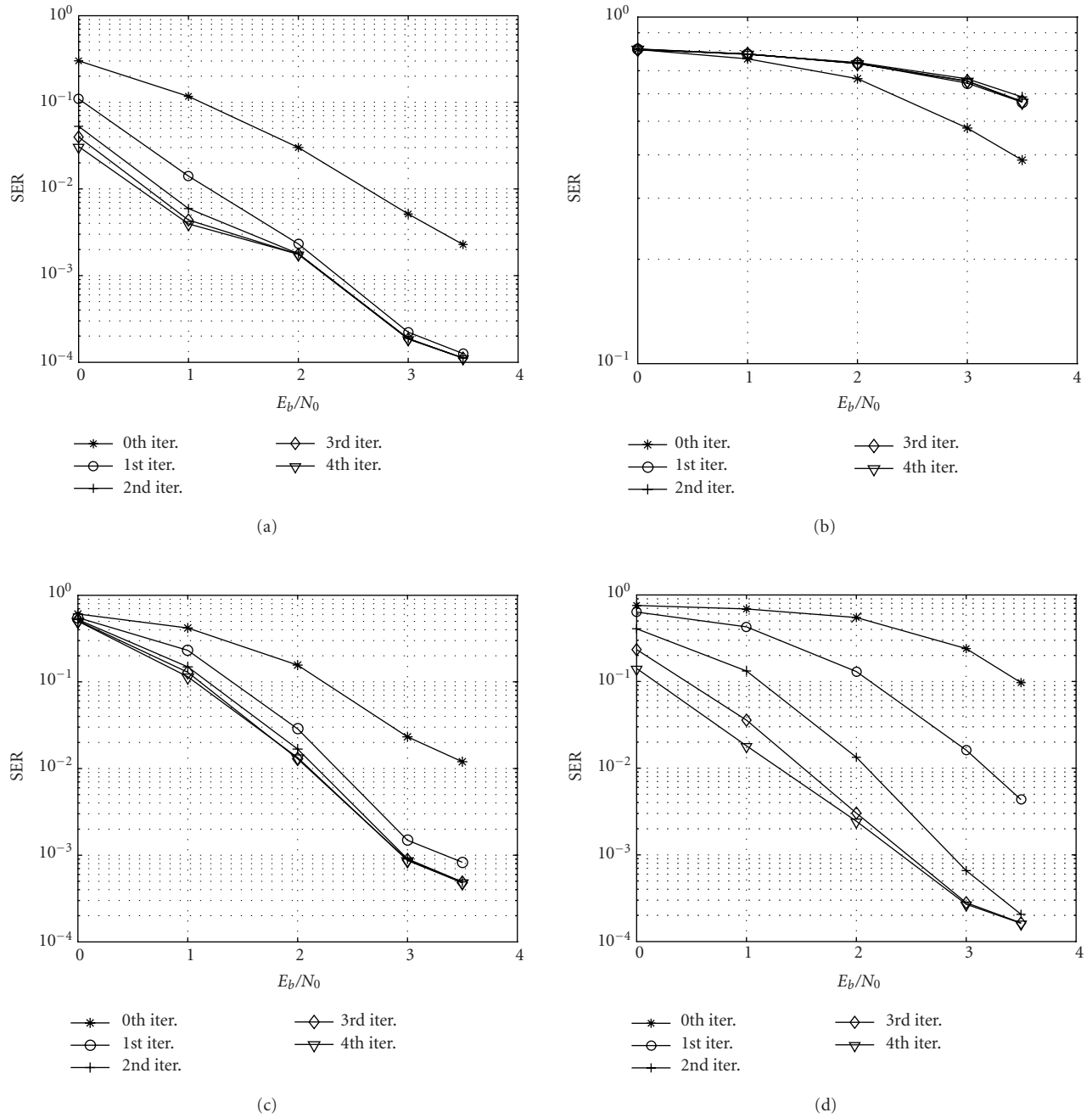


FIGURE 9: SER obtained by iterative source-channel decoding of a Gauss-Markov source quantized on 16 levels and coded with a Huffman code. (a) $\rho = \rho_s = 0.9$; (b) $\rho = 0.9$, $\rho_s = 0.5$; (c) $\rho = 0.5$, $\rho_s = 0.9$; (d) ρ estimated online ($\rho_s = 0.9$).

The performance of this online statistics estimation algorithm is illustrated in Section 8 in the context of JSCD of H.263++ video motion vectors.

8. DISCUSSION AND PERFORMANCE ILLUSTRATIONS

The last years have seen substantial effort beyond the theoretical results and validations on theoretical sources to consider the application of the above techniques in real source coding/decoding systems, for example, for error-resilient transmission of still images and video signals over wireless networks. Among the questions at stake are indeed the viability in practical systems of

(i) SISO source decoding solutions versus hard decoding solutions still very widely used in source decoding systems due to their low decoding complexity;
(ii) JSCD solutions versus the tandem approaches.

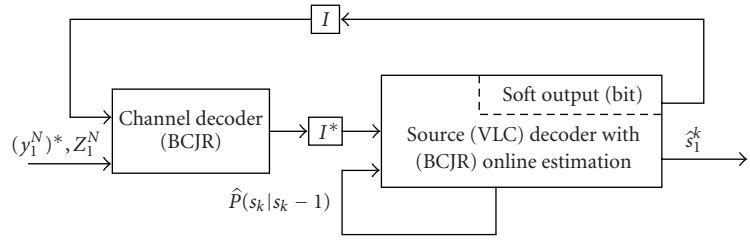


Figure 10: Iterative source-channel decoding with online estimation (courtesy of [87]).

Key factors in relation to these questions are of course SNR performance, complexity, and possibly cross-layer information exchange support.

We first consider the question of benefits of SISO source decoding solutions for state-of-the-art compression systems. As an example, we consider a compression system making use of arithmetic codes which are now the most prominent codes in image and video compression and at the same time the most sensitive to noise. Sequential decoding with soft channel information and soft output has been tested in the JPEG-2000 decoder in [24] together with a soft synchronization technique, making use of the synchronization markers specified in the standard. Figure 11 shows the decoding results with the Lena image encoded at 0.5 bpp and transmitted over an AWGN channel with a signal-to-noise (E_b/N_0) of 5 dB. The standard JPEG-2000 decoder is compared against the sequential decoding technique with an increasing number of surviving paths, corresponding to $W = 10$ and $W = 20$ surviving paths, respectively, that is, to an increasing computational complexity. This shows on one hand the significantly quality gain and on the other hand that the approach allows to flexibly trade estimation reliability (performance) against complexity. This makes this type of approach a viable solution for practical state-of-the-art image compression systems. The authors in [71, 91] show similar benefits of MAP decoding of RVLC- and VLC-encoded texture information in an MPEG-4 video compressed stream. The authors in [92] also apply sequential decoding with both soft and hard channel values to the decoding of startcodes and overhead information in an MPEG-4 video compressed stream. A performance evaluation of MAP and sequential decoding with soft channel information indicates that transmission with no channel coding may be envisaged, provided the Hamming distance between the source codewords is large enough.

We now consider the question of the benefits of JSCD versus tandem decoding solutions. One related question is the form and placement of redundancy: should we maintain a controlled, yet sufficient, amount of redundancy in the source representation? Or should we compress as much as possible the source and use the freed bandwidth for extra channel code redundancy? In relation to this question, one has to bear in mind that, from a source representation and decoding point of view, the quality criterion is, unlike in channel coding, definitely not the *bit error rate*. One single bit error in the entire bitstream can have a dramatic effect on the quality of the reconstructed signal due to the source decoder desynchronization problem. It is thus necessary to dedicate some redundancy to address this specific problem. Many results illustrating this point can be found in the literature with theoretical sources [24, 35].

Here, to illustrate this point, we focus on a set of achievements with real compression systems. The choice of a real compression system is also motivated by the fact that the application of the techniques described above in real video decoding systems raises a certain number of practical issues which deserve to be mentioned. For example, if we consider JSCD of motion vectors, one must account for the fact that the syntax of compressed video stream often multiplexes the horizontal and vertical components of these displacement vectors reducing the dependencies. Motion vectors are also often encoded differentially, reducing the amount of residual correlation. In [89], a joint source-channel decoding technique is used to exploit the residual redundancy between motion vectors in a compressed video stream. The motion vectors are assumed to be ordered so that all the horizontal components are consecutive and then followed by all the vertical displacement components. The authors in [87, 88] proceed similarly with the JSCD of motion vectors in an H.263++ video decoder. The JSCD structure presented in Figure 10 is thus used to decode video sequences encoded according to the H.263++ standard and transmitted over a Rayleigh channel. Figure 12 gives the PSNR values obtained when transmitting the sequence *Flower garden* compressed with H.263+ on a Rayleigh channel. The JSCD system is compared against the tandem structure making use of the channel decoder followed by a hard RVLC decoder. RVLC codes are indeed recommended by the H.263+ standard when using the compressed signals in error-prone environments. The channel coder that has been used in the experiments is an RSC code defined by the polynomials $F(z) = 1 + z^2 + z^3 + z^4$ and $G(z) = 1 + z + z^4$. Note that in the tandem system, the motion vectors are encoded differentially to free some bandwidth used for the redundancy inherent to the RVLC and for the redundancy generated by the channel coder. In the JSCD system, the motion vectors are not encoded in a differential manner. This introduces some form of redundancy in the source that is exploited in a very advantageous way by the iterative decoder. In order to have a comparable overall rate for both systems, in the case of nondifferential encoding, the RSC encoder output is punctured to give a channel code rate of 2/3. The curves reveal a more stable PSNR and a significantly higher average PSNR (gain of 4 dB) for the JSCD approach against the RVLC-RSC structure.

(a) (b)

(c) (d)

Figure 11: Performance of sequential decoding with JPEG-2000 coded images (courtesy of [24]). (a) JPEG-2000 coded; PSNR = 37.41 dB; no channel error. (b) JPEG-2000 coded; AWGN channel ($E_b/N_0 = 5$ dB); PSNR = 16.43 dB. (c) JPEG-2000 coded with sequential decoding; AWGN channel ($E_b/N_0 = 5$ dB); $W = 10$; PSNR = 25.15 dB. (d) JPEG-2000 with sequential decoding; AWGN channel ($E_b/N_0 = 5$ dB); $W = 20$; PSNR = 31.91 dB.

The experiments reported above for the JSCD and the tandem systems make use of a simple convolutional coder. The gain in performance is achieved at the expense of increased complexity. One could consider using a turbo code in the tandem system. This would lead to a complexity comparable to the one of the JSCD chain. Such a comparison between a serial joint source-channel coding/decoding chain and a tandem chain using a parallel turbo code has been made in [93]. It is shown that for low AWGN channel SNR values, JSCD with convolutional codes provides better results than the tandem chain using the parallel turbo code, for the same overall rate. However, for higher channel SNR values, the tandem chain outperforms the serial JSCD system. The study could be pushed further by considering turbo channel codes in both chains, as described in [94]. The joint source-channel decoder thus comprises three SISO modules, one for the VLC decoder and one for each of the RSC constituent decoders of the turbo code. SISO source decoding is not necessarily realized at each iteration, which limits the extra complexity that one could expect for the JSCD chain. The authors in [94] exhibit gains for variable-length encoded images using the JSCD approach based on the three SISO decoders. Although, many issues (e.g., adequation of models to realistic systems, complexity, parameter settings, etc.) still need further investigation, all the above results contribute to illustrate the potential benefit of JSCD for future image and video communication systems.

9. CONCLUSION

This paper has given an overview of recent advances in JSCD of VLC compressed signals. The JSCD framework can be highly beneficial for future wireless multimedia services both because of its higher SER and SNR performance with respect to classical tandem decoding solutions and of the idiosyncrasies of the wireless links. The use of soft source decoding techniques, instead of classical decoding solutions, indeed allows to decrease very significantly the source SER (e.g., 0.4×10^{-2} versus 0.8 for a channel SNR of 3 dB and with arithmetic codes). This SER can be further decreased by using JSCD techniques. Note that the higher performance is however obtained at the expense of an increased complexity, which is an issue which requires further work. Pruning techniques have already been studied in the literature in order to reduce the decoding algorithms' complexity. However, further work is needed, for example, to investigate the

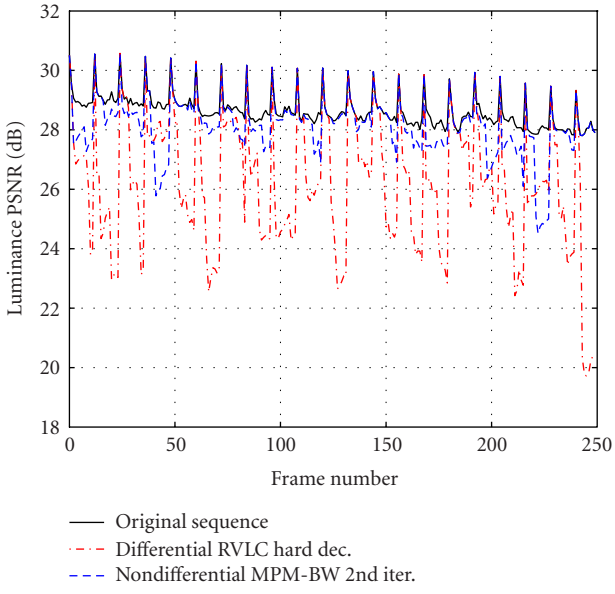


FIGURE 12: PSNR values obtained with the H.263+ compressed "Flower" sequence transmitted over a Rayleigh channel with a tandem scheme using channel decoding followed by hard Huffman decoding with differential MV coding (at the left) and with JSCD using online estimation (at the right) (courtesy of [88]).

respective advantages/drawbacks of bit versus symbol trellises with respect to pruning and complexity reduction, on the best form of redundancy to be introduced in the chain, including the most appropriate resynchronization mechanisms depending on the channel characteristics (random or bursty errors). Also, the implementation of JSCD in practical communication systems optimally requires some vertical cooperation between the application layer and the layers below, with cross-layer soft information exchange. Such ideas of *interlayer* communication which would allow to best select and adapt *subnet* technologies to varying transmission conditions and to application characteristics seem also to be progressing in the networking community [95]. Therefore, before reaching a level of maturity sufficient for a large adoption in standards and practical communication systems, issues such as reduced complexity implementation methods, cross-layer (possibly networked) signaling mechanisms required, and optimal repartition of redundancy between the source and the channel codes still need to be resolved.

ACKNOWLEDGMENTS

Part of this work was carried out when P. Siohan was in a sabbatical leave at INRIA Rennes. The authors would like to thank Dr. Thomas Guionnet, Dr. Claudio Weidmann, and Dr. Marion Jeanne for their help in the preparation of this manuscript. The authors would also like to thank the anonymous reviewers for their very constructive and helpful comments.

REFERENCES

[1] L.-A. Larzon, M. Degermark, S. Pink, L.-E. Jonsson Ed., and G. Fairhurst Ed., "The UDP-lite protocol," IETF internet Draft, December 2002, http://www.ietf.org/proceedings/03jul/I-D/draft-ietf-tsvwg-udp-lite-01.txt.

[2] J. Hagenauer, "Rate-compatible punctured convolutional codes (RCPC codes) and their applications," *IEEE Trans. Commun.*, vol. 36, no. 4, pp. 389–400, 1988.

[3] N. Farvardin, "A study of vector quantization for noisy channels," *IEEE Trans. Inform. Theory*, vol. 36, no. 4, pp. 799–809, 1990.

[4] N. Farvardin and V. Vaishampayan, "On the performance and complexity of channel-optimized vector quantizers," *IEEE Trans. Inform. Theory*, vol. 37, no. 1, pp. 155–160, 1991.

[5] T. J. Ferguson and J. H. Rabinowitz, "Self-synchronizing Huffman codes," *IEEE Trans. Inform. Theory*, vol. 30, no. 4, pp. 687–693, 1984.

[6] W. M. Lam and A. R. Reibman, "Self-synchronizing variable-length codes for image transmission," in *Proc. IEEE Int. Conf. Acoustics, Speech, Signal Processing (ICASSP '92)*, vol. 3, pp. 477–480, San Francisco, Calif, USA, March 1992.

[7] K. Sayood and J. C. Borkenhagen, "Use of residual redundancy in the design of joint source/channel coders," *IEEE Trans. Commun.*, vol. 39, no. 6, pp. 838–846, 1991.

[8] F. Alajaji, N. Phamdo, and T. Fuja, "Channel codes that exploit the residual redundancy in CELP-encoded speech," *IEEE Trans. Speech Audio Processing*, vol. 4, no. 5, pp. 325–336, 1996.

[9] N. Phamdo and N. Farvardin, "Optimal detection of discrete Markov sources over discrete memoryless channels-applications to combined source-channel coding," *IEEE Trans. Inform. Theory*, vol. 40, no. 1, pp. 186–193, 1994.

[10] K. Sayood, F. Liu, and J. D. Gibson, "A constrained joint source/channel coder design," *IEEE J. Select. Areas Commun.*, vol. 12, no. 9, pp. 1584–1593, 1994.

[11] M. Park and D. J. Miller, "Decoding entropy-coded symbols over noisy channels by MAP sequence estimation for asynchronous HMMs," in *Proc. 32nd Annual Conference on Information Sciences and Systems (CISS '98)*, pp. 477–482, Princeton, NJ, USA, March 1998.

[12] J. Wen and J. D. Villasenor, "Utilizing soft information in decoding of variable length codes," in *Proc. IEEE Data Compression Conference (DCC '99)*, pp. 131–139, Snowbird, Utah, USA, March 1999.

[13] M. Park and D. J. Miller, "Joint source-channel decoding for variable-length encoded data by exact and approximate MAP sequence estimation," in *Proc. IEEE Int. Conf. Acoustics, Speech, Signal Processing (ICASSP '99)*, vol. 5, pp. 2451–2454, Phoenix, Ariz, USA, March 1999.

[14] A. Guyader, E. Fabre, C. Guillemot, and M. Robert, "Joint source-channel turbo decoding of entropy-coded sources," *IEEE J. Select. Areas Commun.*, vol. 19, no. 9, pp. 1680–1696, 2001, Special issue on the turbo principle: from theory to practice.

[15] J. Wen and J. Villasenor, "Soft-input soft-output decoding of variable length codes," *IEEE Trans. Commun.*, vol. 50, no. 5, pp. 689–692, 2002.

[16] A. H. Murad and T. E. Fuja, "Joint source-channel decoding of variable-length encoded sources," in *Proc. Information Theory Workshop (ITW '98)*, pp. 94–95, Killarney, Ireland, June 1998.

[17] N. Demir and K. Sayood, "Joint source/channel coding for variable length codes," in *Proc. IEEE Data Compression Conference (DCC '98)*, pp. 139–148, Snowbird, Utah, USA, March–April 1998.

[18] J. Hagenauer and P. Hoeher, "A Viterbi algorithm with soft-decision outputs and its applications," in *Proc. IEEE Global Telecommunications Conference (GLOBECOM '89)*, vol. 3, pp. 1680–1686, Dallas, Tex, USA, November 1989.

[19] L. R. Bahl, J. Cocke, F. Jelinek, and J. Raviv, "Optimal decoding of linear codes for minimizing symbol error rate," *IEEE Trans. Inform. Theory*, vol. 20, no. 2, pp. 284–287, 1974.

[20] V. Buttigieg and P. G. Farrell, "On variable-length error-correcting codes," in *Proc. IEEE International Symposium on Information Theory (ISIT '94)*, p. 507, Trondheim, Norway, June–July 1994.

[21] V. B. Balakirsky, "Joint source-channel coding with variable length codes," in *Proc. IEEE International Symposium on Information Theory (ISIT '97)*, p. 419, Ulm, Germany, June–July 1997.

[22] D. J. Miller and M. Park, "A sequence-based approximate MMSE decoder for source coding over noisy channels using discrete hidden Markov models," *IEEE Trans. Commun.*, vol. 46, no. 2, pp. 222–231, 1998.

[23] B. D. Pettijohn, M. W. Hoffman, and K. Sayood, "Joint source/channel coding using arithmetic codes," *IEEE Trans. Commun.*, vol. 49, no. 5, pp. 826–836, 2001.

[24] T. Guionnet and C. Guillemot, "Soft decoding and synchronization of arithmetic codes: application to image transmission over noisy channels," *IEEE Trans. Image Processing*, vol. 12, no. 12, pp. 1599–1609, 2003.

[25] E. Magli, M. Grangetto, and G. Olmo, "Error correcting arithmetic coding for robust video compression," in *Proc. 6th Baiona Workshop on Signal Processing in Communications*, Baiona, Spain, September 2003, http://www1.tlc.polito.it/SAS/grangetto_pdb.shtml.

[26] G. F. Elmasry, "Embedding channel coding in arithmetic coding," *IEE Proceedings-Communications*, vol. 146, no. 2, pp. 73–78, 1999.

[27] C. Boyd, J. G. Cleary, S. A. Irvine, I. Rinsma-Melchert, and I. H. Witten, "Integrating error detection into arithmetic coding," *IEEE Trans. Commun.*, vol. 45, no. 1, pp. 1–3, 1997.

[28] G. F. Elmasry, "Joint lossless-source and channel coding using automatic repeat request," *IEEE Trans. Commun.*, vol. 47, no. 7, pp. 953–955, 1999.

[29] I. Sodagar, B. B. Chai, and J. Wus, "A new error resilience technique for image compression using arithmetic coding," in *Proc. IEEE Int. Conf. Acoustics, Speech, Signal Processing (ICASSP '00)*, vol. 4, pp. 2127–2130, Istanbul, Turkey, June 2000.

[30] T. Guionnet and C. Guillemot, "Soft and joint source-channel decoding of quasi-arithmetic codes," *Eurasip J. Appl. Signal Process.*, vol. 2004, no. 3, pp. 393–411, 2004.

[31] J. Garcia-Frias and J. D. Villasenor, "Combining hidden Markov source models and parallel concatenated codes," *IEEE Commun. Lett.*, vol. 1, no. 4, pp. 111–113, 1997.

[32] N. Gortz, "On the iterative approximation of optimal joint source-channel decoding," *IEEE J. Select. Areas Commun.*, vol. 19, no. 9, pp. 1662–1670, 2001.

[33] M. Adrat, U. von Agris, and P. Vary, "Convergence behavior of iterative source-channel decoding," in *Proc. IEEE Int. Conf. Acoustics, Speech, Signal Processing (ICASSP '03)*, vol. 4, pp. 269–272, Hong Kong, China, April 2003.

[34] S. ten Brink, "Convergence behavior of iteratively decoded parallel concatenated codes," *IEEE Trans. Commun.*, vol. 49, no. 10, pp. 1727–1737, 2001.

[35] R. Bauer and J. Hagenauer, "Iterative source/channel- decoding using reversible variable length codes," in *Proc. IEEE Data Compression Conference (DCC '00)*, pp. 93–102, Snowbird, Utah, USA, March 2000.

[36] R. Bauer and J. Hagenauer, "Turbo FEC/VLC decoding and its application to text compression," in *Proc. 34th Annual Conference on Information Sciences and Systems (CISS '00)*, pp. WA6.6–WA6.11, Princeton, NJ, USA, March 2000.

[37] R. Bauer and J. Hagenauer, "Symbol-by-symbol MAP decoding of variable length codes," in *Proc. 3rd ITG Conference on Source and Channel Coding (CSCC '00)*, pp. 111–116, Munich, Germany, January 2000.

[38] J. Kliewer and R. Thobaben, "Parallel concatenated joint source-channel coding," *Electronics Letters*, vol. 39, no. 23, pp. 1664–1666, 2003.

[39] J. Hagenauer, "Source-controlled channel decoding," *IEEE Trans. Commun.*, vol. 43, no. 9, pp. 2449–2457, 1995.

[40] J. Garcia-Frias and J. D. Villasenor, "Joint turbo decoding and estimation of hidden Markov sources," *IEEE J. Select. Areas Commun.*, vol. 19, no. 9, pp. 1671–1679, 2001.

[41] G.-C. Zhu, F. Alajaji, J. Bajcsy, and P. Mitran, "Non-systematic turbo codes for non-uniform i.i.d. sources over AWGN channels," in *Proc. Conference on Information Sciences and Systems (CISS '02)*, Princeton, NJ, USA, March 2002.

[42] L. Guivarch, J.-C. Carlach, and P. Siohan, "Joint source-channel soft decoding of Huffman codes with turbo-codes," in *Proc. IEEE Data Compression Conference (DCC '00)*, pp. 83–92, Snowbird, Utah, USA, March 2000.

[43] M. Jeanne, J.-C. Carlach, and P. Siohan, "Joint source-channel decoding of variable-length codes for convolutional codes and turbo codes," *IEEE Trans. Commun.*, vol. 53, no. 1, pp. 10–15, 2005.

[44] G. D. Forney, "The Viterbi algorithm," *Proc. IEEE*, vol. 61, no. 3, pp. 268–278, 1973.

[45] R. W. Chang and J. C. Hancock, "On receiver structures for channels having memory," *IEEE Trans. Inform. Theory*, vol. 12, no. 4, pp. 463–468, 1966.

[46] P. L. McAdam, L. Welch, and C. Weber, "MAP bit decoding of convolutional codes," in *Proc. IEEE International Symposium on Information Theory (ISIT '72)*, Asilomar, Calif, USA, January 1972.

[47] J. A. Erfanian and S. Pasupathy, "Low-complexity parallel-structure symbol-by-symbol detection for ISI channels," in *Proc. IEEE Pacific Rim Conference on Communications, Computers and Signal Processing*, pp. 350–353, Victoria, BC, Canada, June 1989.

[48] P. Robertson, E. Villebrun, and P. Hoeher, "A comparison of optimal and sub-optimal MAP decoding algorithms operating in the log domain," in *Proc. IEEE International Conference on Communications (ICC '95)*, vol. 2, pp. 1009–1013, Seattle, Wash, USA, June 1995.

[49] F. R. Kschischang and B. J. Frey, "Iterative decoding of compound codes by probability propagation in graphical models," *IEEE J. Select. Areas Commun.*, vol. 16, no. 2, pp. 219–230, 1998.

[50] D. A. Huffman, "A method for the construction of minimum redundancy codes," *Proc. IRE*, vol. 40, no. 9, pp. 1098–1101, 1952.

[51] Y. Takishima, M. Wada, and H. Murakami, "Reversible variable length codes," *IEEE Trans. Commun.*, vol. 43, no. 2/3/4, pp. 158–162, 1995.

[52] R. Bauer and J. Hagenauer, "On variable length codes for iterative source/channel decoding," in *Proc. IEEE Data Compression Conference (DCC '01)*, pp. 273–282, Snowbird, Utah, USA, March 2001.

[53] M. Jeanne, J.-C. Carlach, P. Siohan, and L. Guivarch, "Source and joint source-channel decoding of variable length codes," in *Proc. IEEE International Conference on Communications (ICC '02)*, vol. 2, pp. 768–772, New York, NY, USA, April–May 2002.

[54] K. P. Subbalakshmi and J. Vaisey, "Joint source-channel decoding of entropy coded Markov sources over binary symmetric channels," in *Proc. IEEE International Conference on Communications (ICC '99)*, vol. 1, pp. 446–450, Vancouver, BC, Canada, June 1999.

[55] J. J. Rissanen, "Arithmetic codings as number representations," *Acta Polytechnica Scandinavica*, vol. 31, pp. 44–51, 1979.

[56] P. G. Howard and J. S. Vitter, "Practical implementations of arithmetic coding," in *Image and Text Compression*, J. A. Storer, Ed., pp. 85–112, Kluwer Academic Publishers, Norwell, Mass, USA, 1992.

[57] P. G. Howard and J. S. Vitter, "Design and analysis of fast text compression based on quasi-arithmetic coding," in *Proc. IEEE Data Compression Conference (DCC '93)*, pp. 98–107, Snowbird, Utah, USA, March–April 1993.

[58] D. Taubman, "High performance scalable image compression with EBCOT," *IEEE Trans. Image Processing*, vol. 9, no. 7, pp. 1158–1170, 2000.

[59] D. Marpe, G. Blättermann, G. Heising, and T. Wiegand, "Video compression using context-based adaptive arithmetic coding," in *Proceedings of IEEE International Conference on Image Processing (ICIP '01)*, vol. 3, pp. 558–561, Thessaloniki, Greece, October 2001.

[60] C. Christopoulos, A. Skodras, and T. Ebrahimi, "The JPEG2000 still image coding system: an overview," *IEEE Trans. Consumer Electron.*, vol. 46, no. 4, pp. 1103–1127, 2000, ISO/IEC JTC1/SC29/WG1 (ITU-T) SG8.

[61] T. Wiegand and G. Sullivan, "Draft ISO/IEC 14496-10 AVC," March 2003, http://www.h263l.com/h264/JVT-G050.pdf.

[62] M. P. C. Fossorier, F. Burkert, S. Lin, and J. Hagenauer, "On the equivalence between SOVA and max-log-MAP decodings," *IEEE Commun. Lett.*, vol. 2, no. 5, pp. 137–139, 1998.

[63] L. Gong, W. Xiaofu, and Y. Xiaoxin, "On SOVA for nonbinary codes," *IEEE Commun. Lett.*, vol. 3, no. 12, pp. 335–337, 1999.

[64] J. Tan and G. L. Stuber, "A MAP equivalent SOVA for nonbinary turbo codes," in *Proc. IEEE International Conference on Communications (ICC '00)*, vol. 2, pp. 602–606, New Orleans, La, USA, June 2000.

[65] A. J. Viterbi, "An intuitive justification and a simplified implementation of the MAP decoder for convolutional codes," *IEEE J. Select. Areas Commun.*, vol. 16, no. 2, pp. 260–264, 1998.

[66] R. M. Fano, "A heuristic discussion of probabilistic decoding," *IEEE Trans. Inform. Theory*, vol. 9, no. 2, pp. 64–74, 1963.

[67] S. Lin and D. J. Costello Jr., *Error Control Coding: Fundamentals and Applications*, Prentice Hall, Englewood Cliffs, NJ, USA, 1983.

[68] J. L. Massey, "Variable-length codes and the Fano metric," *IEEE Trans. Inform. Theory*, vol. 18, no. 1, pp. 196–198, 1972.

[69] C. Weiss, S. Riedel, and J. Hagenauer, "Sequential decoding using a priori information," *Electronics Letters*, vol. 32, no. 13, pp. 1190–1191, 1996.

[70] A. Kopansky, *Joint source-channel decoding for robust transmission of video*, Ph.D. thesis, Drexel University, Philadelphia, Pa, USA, August 2002.

[71] L. Perros-Meilhac and C. Lamy, "Huffman tree based metric derivation for a low-complexity sequential soft VLC decoding," in *Proc. IEEE International Conference on Communications (ICC '02)*, vol. 2, pp. 783–787, New York, NY, USA, April–May 2002.

[72] C. Lamy and L. Perros-Meilhac, "Low complexity iterative decoding of variable-length codes," in *Proc. Picture Coding Symposium (PCS '03)*, pp. 275–280, Saint Malo, France, April 2003.

[73] F. Jelinek, "Fast sequential decoding algorithm using a stack," *IBM Journal of Research and Development*, vol. 13, no. 6, pp. 675–685, 1969.

[74] C. Demiroglu, M. W. Hoffman, and K. Sayood, "Joint source channel coding using arithmetic codes and trellis coded modulation," in *Proc. IEEE Data Compression Conference (DCC '01)*, pp. 302–311, Snowbird, Utah, USA, March 2001.

[75] C. Berrou and A. Glavieux, "Near optimum error correcting coding and decoding: turbo-codes," *IEEE Trans. Commun.*, vol. 44, no. 10, pp. 1261–1271, 1996.

[76] B. J. Frey and D. J. C. MacKay, "A revolution: belief propagation in graphs with cycles," in *Proc. Neural Information Processing Systems Conference (NIPS '97)*, Denver, Colo, USA, December 1997.

[77] R. J. McEliece, D. J. C. MacKay, and J.-F. Cheng, "Turbo decoding as an instance of Pearl's "belief propagation" algorithm," *IEEE J. Select. Areas Commun.*, vol. 16, no. 2, pp. 140–152, 1998.

[78] J. Hagenauer, "The turbo principle: tutorial introduction and state of the art," in *Proc. International Symposium on Turbo Codes and Related Topics*, pp. 1–11, Brest, France, September 1997.

[79] M. Jeanne, *Etude des systèmes robustes de décodage conjoint source-canal pour la transmission sans fil de vidéo*, Ph.D. thesis, Instituts Nationaux des Sciences Appliquées (INSA), Rennes, France, 2003.

[80] S. Benedetto, G. Montorsi, D. Divsalar, and F. Pollara, "Iterative decoding of serially concatenated codes with interleavers and comparison with turbo codes," in *Proc. IEEE Global Telecommunications Conference (GLOBECOM '97)*, vol. 2, pp. 654–658, Phoenix, Ariz, USA, November 1997.

[81] G.-C. Zhu and F. Alajaji, "Turbo codes for nonuniform memoryless sources over noisy channels," *IEEE Commun. Lett.*, vol. 6, no. 2, pp. 64–66, 2002.

[82] A. Elbaz, R. Pyndiah, B. Solaiman, and O. Ait Sab, "Iterative decoding of product codes with a priori information over a Gaussian channel for still image transmission," in *Proc. IEEE Global Telecommunications Conference (GLOBECOM '99)*, vol. 5, pp. 2602–2606, Rio de Janeireo, Brazil, December 1999.

[83] Z. Peng, Y.-F. Huang, and D. J. Costello Jr., "Turbo codes for image transmission-a joint channel and source decoding approach," *IEEE J. Select. Areas Commun.*, vol. 18, no. 6, pp. 868–879, 2000.

[84] L. Guivarch, P. Siohan, and J.-C. Carlach, "Low complexity soft decoding of Huffman encoded Markov sources using turbo-codes," in *Proc. IEEE 7th International Conference on Telecommunications (ICT '00)*, pp. 872–876, Acapulco, Mexico, May 2000.

[85] T. Okuda, E. Tanaka, and T. Kasai, "A method for the correction of garbled words based on the Levenshtein metric," *IEEE Trans. Comput.*, vol. 25, no. 2, pp. 172–178, 1976.

[86] K. Lakovic and J. Villasenor, "Combining variable length codes and turbo codes," in *Proc. IEEE 55th Vehicular Technology Conference (VTC '02)*, vol. 4, pp. 1719–1723, Birmingham, Ala, USA, May 2002.

[87] C. Weidmann and P. Siohan, "Décodage conjoint source-canal avec estimation en ligne de la source," in *Compression et Représentation des Signaux Audiovisuels (CORESA '03)*, Lyon, France, January 2003.

[88] C. Weidmann and P. Siohan, "Vidéo sur canal radio-mobile," Tech. Rep. 59, 2003, Chapter 8 of COSOCATI RNRT, http://www.telecom.gouv.fr/rnrt/rnrt/projets/res_d59_ap99.htm.

[89] A. H. Murad and T. E. Fuja, "Robust transmission of variable-length encoded sources," in *Proc. IEEE Wireless Communications and Networking Conference (WCNC '99)*, vol. 2, pp. 968–972, New Orleans, La, USA, September 1999.

[90] L. R. Rabiner, "A tutorial on hidden Markov models and selected applications in speech recognition," *Proc. IEEE*, vol. 77, no. 2, pp. 257–286, 1989.

[91] K. P. Subbalakshmi and Q. Chen, "Joint source-channel decoding for MPEG-4 coded video over wireless channels," in *Proc. IASTED International Conference on Wireless and Optical Communications (WOC '02)*, pp. 617–622, Banff, AB, Canada, July 2002.

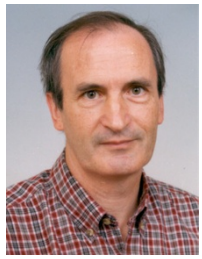
[92] A. Kopansky and M. Bystrom, "Sequential decoding of MPEG-4 coded bitstreams for error resilience," in *Proc. 33rd Annual Conference on Information Sciences and Systems (CISS '99)*, Baltimore, Md, USA, March 1999.

[93] M. Jeanne, P. Siohan, and J.-C. Carlach, "Comparaison de deux approches du décodage conjoint source-canal," in *Proc. GRETSI*, Paris, France, September 2003.

[94] X. Jaspar and L. Vandendorpe, "Three SISO modules joint source-channel turbo decoding of variable length coded images," in *Proc. 5th International ITG Conference on Source and Channel Coding (SCC '04)*, Erlangen, Germany, January 2004.

[95] S. Mérigeault and C. Lamy, "Concepts for exchanging extra information between protocol layers transparently for the standard protocol stack," in *Proc. IEEE 10th International Conference on Telecommunications (ICT '03)*, vol. 2, pp. 981–985, Tahiti, French Polynesia, February–March 2003.

Christine Guillemot is currently "Directeur de Recherche" at INRIA, in charge of a research group dealing with image modelling, processing, and video communication. She holds a Ph.D. degree from ENST (Ecole Nationale Supérieure des Télécommunications), Paris. From 1985 to October 1997, she has been with France Telecom/CNET, where she has been involved in various projects in the domain of coding for TV, HDTV, and multimedia applications. From January 1990 to mid 1991, she worked at Bellcore, NJ, USA, as a Visiting Scientist. Her research interests are in signal and image processing, video coding, and joint source and channel coding for video transmission over the Internet and over wireless networks. She is a Member of the IEEE IMDSP Committee. She served as an Associated Editor for IEEE Transactions on Image Processing (2000–2003) and is currently an Associated Editor for the IEEE Transactions on Circuits and Systems for Video Technology.



Pierre Siohan was born in Camlez, France, in October 1949. He received the Ph.D. degree from the École Nationale Supérieure des Télécommunications (ENST), Paris, France, in 1989, and the Habilitation degree from the University of Rennes, Rennes, France, in 1995. In 1977, he joined the Centre Commun d'Études de Télédiffusion et Télécommunications (CCETT), Rennes, where his activities were first concerned with the communication theory and its application to the design of broadcasting systems. Between 1984 and 1997, he was in charge of the Mathematical and Signal Processing Group. Since September 1997, he has been an Expert Member in the R&D Division of France Télécom working in the Broadband Wireless Access Laboratory. From September 2001 to September 2003, he took a two-year sabbatical leave, being a Directeur de Recherche at the Institut National de Recherche en Informatique et Automatique (INRIA), Rennes. His current research interests are in the areas of filter-bank design for communication systems, joint source-channel coding, and distributed source coding.