Progressive and Error-Resilient Transmission Strategies for VLC Encoded Signals over Noisy Channels

Hervé Jégou, Christine Guillemot

Hindawi Publishing Corporation
EURASIP Journal on Applied Signal Processing
Volume 2006, Article ID 37164, Pages 1–18
DOI 10.1155/ASP/2006/37164

Progressive and Error-Resilient Transmission Strategies for VLC Encoded Signals over Noisy Channels

Hervé Jégou[1] and Christine Guillemot[2]

[1] *IRISA, Université de Rennes, Campus Universitaire de Beaulieu, 35042 Rennes, France*
[2] *INRIA Rennes IRISA, Campus Universitaire de Beaulieu, 35042 Rennes, France*

Received 1 March 2005; Revised 10 August 2005; Accepted 1 September 2005

This paper addresses the issue of robust and progressive transmission of signals (e.g., images, video) encoded with variable length codes (VLCs) over error-prone channels. This paper first describes bitstream construction methods offering good properties in terms of error resilience and progressivity. In contrast with related algorithms described in the literature, all proposed methods have a linear complexity as the sequence length increases. The applicability of soft-input soft-output (SISO) and turbo decoding principles to resulting bitstream structures is investigated. In addition to error resilience, the amenability of the bitstream construction methods to progressive decoding is considered. The problem of code design for achieving good performance in terms of error resilience and progressive decoding with these transmission strategies is then addressed. The VLC code has to be such that the symbol *energy* is mainly concentrated on the first bits of the symbol representation (i.e., on the first transitions of the corresponding codetree). Simulation results reveal high performance in terms of symbol error rate (SER) and mean-square reconstruction error (MSE). These error-resilience and progressivity properties are obtained without any penalty in compression efficiency. Codes with such properties are of strong interest for the binarization of M-ary sources in state-of-the-art image, and video coding systems making use of, for example, the EBCOT or CABAC algorithms. A prior statistical analysis of the signal allows the construction of the appropriate binarization code.

1. INTRODUCTION

Entropy coding, producing VLC, is a core component of any image and video compression scheme. The main drawback of VLCs is their high sensitivity to channel noise: when some bits are altered by the channel, synchronization losses can occur at the receiver, the positions of symbol boundaries are not properly estimated, leading to dramatic symbol error rates (SER). This phenomenon has motivated studies of the synchronization capability of VLCs as well as the design of codes with better synchronization properties [1–3]. Reversible VLCs [4–6] have also been designed to fight against desynchronizations. Soft VLC decoding ideas, exploiting residual source redundancy (the so-called "excess-rate") as well as the intersymbol dependency, have also been shown to reduce the "de-synchronization" effect [7, 8].

For a given number of source symbols, the number of bits produced by a VLC coder is a random variable. The decoding problem is then to properly segment the noisy bitstream into measures on symbols and to estimate the symbols from the noisy sequence of bits (or measurements) that is received. This segmentation problem can be addressed by introducing a priori information in the bitstream, taking often the form of synchronization patterns. This a priori information is then exploited to formulate constraints. One can alternatively, by properly structuring the bitstream, reveal and exploit constraints on some bit positions [9]. A structure of fixed length size slots inherently creates hard synchronization points in the bitstream. The resulting bitstream structure is called error-resilient entropy codes (EREC). The principle can however be pushed further in order to optimize the criteria of resilience, computational complexity, and progressivity.

In this paper, given a VLC, we first focus on the design of transmission schemes of the codewords in order to achieve high SER and signal-to-noise ratio (SNR) performance in the presence of transmission errors. The process for constructing the bitstream is regarded as a dynamic bit mapping between an intermediate binary representation of the sequence of symbols and the bitstream to be transmitted on the channel. The intermediate representation is obtained by assigning codewords to the different symbols. The decoder proceeds similarly with a bit mapping which, in the presence of transmission noise, may not be the inverse of the mapping realized on the sender side, leading to potential decoder desynchronization. The mapping can also be regarded as the

construction of a new VLC for the entire sequence of symbols. Maximum error resilience is achieved when the highest number of bit mappings (performed by coder and decoder) are deterministic. *Constant* and *stable* mappings with different synchronization properties in the presence of transmission errors are introduced. This general framework leads naturally to several versions of the transmission scheme, exploiting the different mapping properties. By contrast with the EREC algorithm, all proposed algorithms have a linear complexity as the sequence length increases. The bitstream construction methods presented lead to significant improvements in terms of SER and SNR with respect to classical transmission schemes, where the variable length codewords are simply concatenated. The proposed approach may be coupled with other complementary approaches of the literature [10]. In particular, granted that the channel properties are known, unequal error protection schemes using rate compatible punctured codes (RCPC) [11] or specific approaches such as [12] may be used to improve the error resilience.

Another design criterion that we consider is the amenability of the VLCs and of transmission schemes for progressive decoding. The notion of progressive decoding is very important for image, video, and audio applications. This is among the features that have been targeted in the embedded stream representation existing in the JPEG2000 standard. For this purpose, an expectation-based decoding procedure is introduced. In order to obtain best progressive SNR performance in the presence of transmission errors, the VLC codetrees have to be designed in such a way that most of the symbols *energy* are concentrated on transitions on the codetree corresponding to bits that will be mapped in a deterministic way. Given a VLC tree (e.g., a Huffman codetree [13]), one can build a new codetree by reassigning the codewords to the different symbols in order to satisfy at best the above criterion, while maintaining the same expected description length (edl) for the corresponding source. This leads to codes referred to as *pseudolexicographic* codes. The lexicographic order can also be enforced by the mean of the Hu-Tucker algorithm [14]. This algorithm returns the lexicographic code having the smallest edl. Among potential applications of these codes, one can cite the error-resilient transmission of images and videos, or the binarization step of coding algorithms such as EBCOT [15] and CABAC [16] used in state-of-the-art image and video coders/decoders. A prior analysis of the statistical distributions of the signals to be encoded (e.g., wavelet coefficients, residue signals) allows the design of binarization codes with appropriate properties of energy compaction on the first transitions of the codetree. This in turn leads to higher mean-square error (MSE) decrease when decoding the first bit-planes (or bins) transmitted.

The rest of the paper is organized as follows. Section 2 introduces the framework of bitstream construction, the notations and definitions used. Several bitstream construction methods offering different trade-offs in terms of error resilience and complexity are described in Section 3. The application of the SISO and the turbo decoding principles to the bitstream resulting from constant mapping is described in Section 4. The code design is discussed in Section 5 and some choices are advocated. Simulation results are provided and discussed in Section 6. The performance of the bitstream construction methods and of the codes are also assessed with a simple image coder. The amenability of VLCs to be used as a binarization tool for modern video coders is discussed.

2. PROBLEM STATEMENT AND DEFINITIONS

Let $\mathbf{S} = (S_1, \ldots, S_t, \ldots, S_K)$ be a sequence of source symbols taking their values in a finite alphabet $\mathcal{A}$ composed of $|\mathcal{A}|$ symbols, $\mathcal{A} = \{a_1, \ldots, a_i, \ldots, a_{|\mathcal{A}|}\}$. These symbols can be wavelet or other transform coefficients which have been quantized. Let $\mathcal{C}$ be a binary variable length code designed for this alphabet, according to its stationary probability $\boldsymbol{\mu} = \{\mu_1, \ldots, \mu_i, \ldots, \mu_{|\mathcal{A}|}\}$. To each symbol S_t is associated a codeword $\mathcal{C}(S_t) = B_t^1 \cdots B_t^{L(S_t)}$ of length $L(S_t)$. The sequence of symbols $\mathbf{S}$ is converted into an intermediate representation,

$$\mathbf{B} = (\mathbf{B_1}; \ldots ; \mathbf{B_K}), \tag{1}$$

where $\mathbf{B_t}$ is a column vector defined as

$$\mathbf{B_t} = \begin{pmatrix} B_t^1 \\ \vdots \\ B_t^{L(S_t)} \end{pmatrix}. \tag{2}$$

In what follows, the emitted bitstream is denoted $\mathbf{E} = E_1 \cdots E_{K_E}$ and the received sequence of noisy bits is denoted $\hat{\mathbf{E}} = \hat{E}_1 \cdots \hat{E}_{K_E}$. Similarly, the intermediate representation on the receiver side is referred to as $\hat{\mathbf{B}}$.

We consider the general framework depicted in Figure 1, where the coding process is decomposed into two steps: codeword assignment (*CA*) and bitstream construction (*BC*). In classical compression systems, the codewords produced are transmitted *sequentially*, forming a *concatenated* bitstream. Here, we focus on the problem of designing algorithms for constructing bitstreams that will satisfy various properties of resiliency and progressivity. Note that both the sequence length K and the length $K_E = \sum_{t=1}^{K} L(S_t)$ of the constructed bitstream $\mathbf{E}$ are assumed to be known on the decoder side. Note also that we reserve capital letters to represent random variables. Small letters will be used to denote the values or realizations of these variables.

The *BC* algorithms can be regarded as dynamic bit *mappings* between the intermediate representation $\mathbf{B}$ and the bitstream $\mathbf{E}$. These mappings φ are thus defined on the set $\mathcal{I}(\mathbf{b}) = \{(t,l)/1 \le t \le K,\ 1 \le l \le L(s_t)\}$ of tuples (t,l) that parses $\mathbf{b}$ (the realization of $\mathbf{B}$) as

$$\begin{aligned} \mathcal{I}(\mathbf{b}) &\longrightarrow [1 \cdots K_E], \\ (t,l) &\longmapsto \varphi(t,l) = n, \end{aligned} \tag{3}$$

where n stands for a bit position of $\mathbf{E}$. Note that the index l can be regarded as the index of a layer (or a bit-plane) in the coded representation of the symbol. Similarly, the decoder proceeds with a bit mapping between the received bitstream

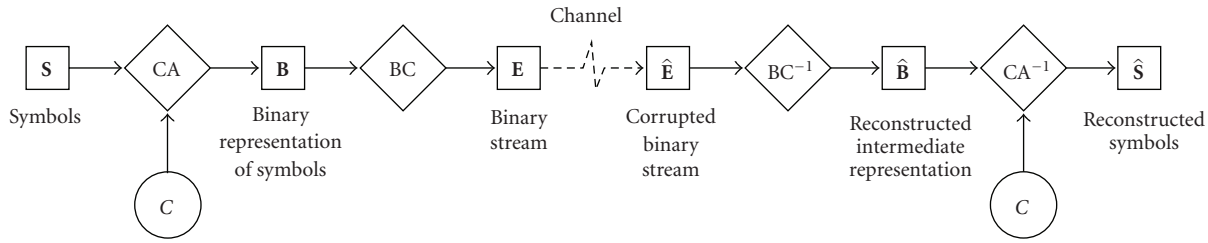

FIGURE 1: Coding and decoding building blocks with the code $\mathcal{C}$: codeword assignment (CA) and bitstream construction (BC).

$\hat{\mathbf{E}}$ and an intermediate representation $\hat{\mathbf{B}}$ of the received sequence of codewords. This mapping, referred to as ψ, depends on the noisy realization $\hat{\mathbf{b}}$ of $\hat{\mathbf{B}}$ and is defined as

$$\begin{aligned}[1\cdots K_E] &\longrightarrow \mathcal{I}(\hat{\mathbf{b}}),\\ n &\longmapsto \psi(n) = (t,l),\end{aligned} \tag{4}$$

where the set $\mathcal{I}(\hat{\mathbf{b}})$, in presence of bit errors, may not be equal to $\mathcal{I}(\mathbf{b})$. The composed function $\pi = \psi \circ \varphi$ is a dynamic mapping function from $\mathcal{I}(\mathbf{b})$ into $\mathcal{I}(\hat{\mathbf{b}})$. An element is decoded in the correct position if and only if $\pi(t,l) = (t,l)$. The error resilience depends on the capability, in presence of channel errors, to map a bitstream element n of $\hat{\mathbf{E}}$ to the correct position (t,l) in the intermediate representation $\hat{\mathbf{B}}$ on the receiver side.

Definition 1. An element index (t,l) is said to be *constant* by $\pi = \psi \circ \varphi$ if and only if $n = \varphi(t,l)$ does not depend on the realization $\mathbf{b}$. Similarly, the bitstream index n is also said to be *constant*.

Let $\mathcal{I}_C$ denote the set of *constant* indexes. The restriction φ_C of φ to the definition set $\mathcal{I}_C$ and its inverse $\psi_C = \varphi_C^{-1}$ are also said to be *constant*. Such constant mappings cannot be altered by channel noise: for all $\hat{\mathbf{b}}, (t,l) \in \mathcal{I}_C \Rightarrow \pi(t,l) = (t,l)$. Let $h_{\mathcal{C}}^-$ and $h_{\mathcal{C}}^+$ denote the length of the shortest and of the longest codewords of the codetree, respectively.

Definition 2. An element index (t,l) is said to be *stable* by π if and only if $\varphi(t,l)$ only depends on $B_t^1, \ldots, B_t^{l-1}$ and for all $l'/1 \le l' < l$, (t,l') is *stable*.

Let $\mathcal{I}_S$ denote the set of *stable* indexes. A stable mapping φ_S can be defined by restricting the mapping φ to the definition set $\mathcal{I}_S$. For the set of stable indexes, the error propagation is restricted to the symbol itself.

Let us consider the transmission of a VLC encoded source on a binary symmetric channel (BSC) with a bit error rate (BER) p. Provided that there is no intersymbol dependence, the probability that a symbol S_t is correctly decoded is given by $P(\hat{S}_t = s_t \mid S_t = s_t) = (1-p)^{L(S_t)}$, leading to the following SER bound:

$$\mathrm{SER}_{\mathrm{bound}}(\mathcal{C}) = 1 - \sum_{a_i \in \mathcal{A}} \mu_i (1-p)^{L(a_i)} = p h_{\mathcal{C}} + \mathcal{O}(p^2), \tag{5}$$

where $h_{\mathcal{C}}$ denotes the edl of the code $\mathcal{C}$. This equation provides a lower bound in terms of SER when transmitting sources encoded with the code $\mathcal{C}$ on a BSC, assuming that simple hard decoding is used. Note that this bound is lower than the SER that would be achieved with fixed length codes (FLCs).

3. BITSTREAM CONSTRUCTION ALGORITHMS

In this section, we describe practical bitstream construction algorithms offering different trade-offs in terms of error resilience and complexity.

3.1. Constant mapping (CMA)

Given a code $\mathcal{C}$, the first approach maximizes the cardinal of the definition set of the constant mapping φ_C, that is, such that $\mathcal{I}_C = [1\cdots K] \times [1\cdots h_{\mathcal{C}}^-]$. Notice first that a variable length codetree comprises a section of a fixed length equal to the minimum length of a codeword denoted $h_{\mathcal{C}}^-$, followed by a variable length section. A *constant* mapping can thus be defined as the composition of functions $\varphi_C : [1\cdots K] \times [1\cdots h_{\mathcal{C}}^-] \to [1\cdots K h_{\mathcal{C}}^-]$ and $\psi_C = \varphi_C^{-1}$ defined such that

$$(t,l) \longrightarrow \varphi_C(t,l) = (l-1)K + t. \tag{6}$$

The bits that do not belong to the definition set of φ_C can be simply concatenated at the end of the bitstream. The *constant* mapping φ_C defines a set of "hard" synchronization points.

Example 1. Let $\mathcal{A} = \{a_1, a_2, a_3, a_4, a_5\}$ be the alphabet of the source $\mathbf{S}_{(1)}$ with the stationary probabilities given by $\mu_1 = 0.4$, $\mu_2 = 0.2$, $\mu_3 = 0.2$, $\mu_4 = 0.1$, and $\mu_5 = 0.1$. This source has been considered by several authors, for example, in [1, 17]. The codes referred to as $\mathcal{C}_5 = \{01, 00, 11, 100, 101\}$ and $\mathcal{C}_7 = \{0, 10, 110, 1110, 1111\}$ in [17] are considered here. The realization $\mathbf{s} = a_1a_4a_5a_2a_3a_3a_1a_2$ leads to the sequence length $K_E = 18$ for code $\mathcal{C}_5$ and to the sequence length $K_E = 20$ for code $\mathcal{C}_7$, respectively. The respective intermediate representations associated to the sequence of symbols $\mathbf{s}$ are given in Figure 2. The CMA algorithm proceeds with the mapping φ of the elements (b_t^l) to, respectively, positions $n = 1\cdots 18$ and $n = 1\cdots 20$ of the bitstream, as illustrated in Figure 2. This finally leads to the bitstreams $\mathbf{e} = 011011001000111001$ and $\mathbf{e} = 011111101\ 110111010100$ for $\mathcal{C}_5$ and $\mathcal{C}_7$, respectively. Note that the set $\mathcal{I}_C$ of constant indexes associated with $\mathcal{C}_5$ is $[1\cdots 8] \times [1;2]$ and with $\mathcal{C}_7$ is $[1\cdots 8] \times [1;1]$.

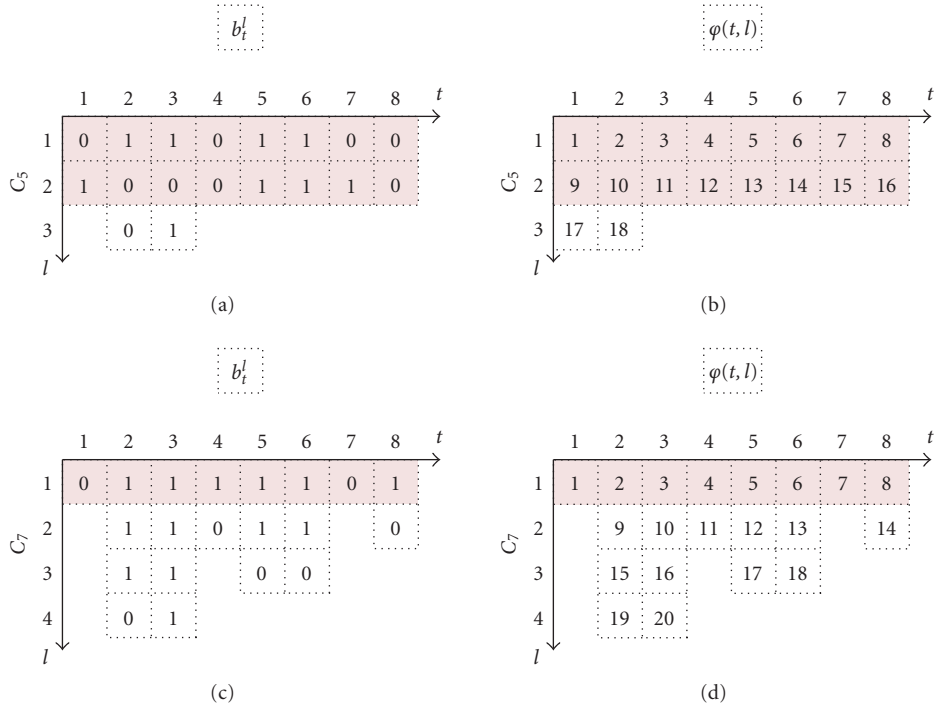

Figure 2: Example of intermediate representation **b** and corresponding mapping φ realized by the CMA algorithm. The set $\mathcal{I}_C$ of constant element indexes is highlighted in gray.

Error propagation will only take place on the tuples (t,l) which do not belong to $\mathcal{I}_C$. The above mapping means transmitting the fixed length section of the codewords bit-plane per bit-plane. Hence, for a Huffman tree, the most frequent symbols will not suffer from desynchronization.

3.2. Stable mapping (SMA) algorithm

The CMA algorithm maximizes the cardinal of the definition set $\mathcal{I}_C$. The error resilience can be further increased by trying to maximize the number of *stable positions*, that is, by minimizing the number of intersymbol dependencies, according to Definition 2. The stability property can be guaranteed for a set $\mathcal{I}_S$ of element indexes (t,l) defined as $\mathcal{I}_S = \{(t,l) \in \mathcal{I}(\mathbf{b})/1 \le t \le K, 1 \le l \le l_s\} \cup \{(t,l) \in \mathcal{I}(\mathbf{b})/1 \le t \le K_s, l = l_s + 1\}$ for l_s and K_s satisfying $l_s \times K + K_s \le K_E$. The expectation of $|\mathcal{I}_S|$ will be maximized by choosing $l_s = \lfloor K_E/K \rfloor$ and $K_s = K_E \mod K$. Let us remind that $\mathcal{I}(\mathbf{b})$ is the definition set of the realization **b** of the intermediate representation **B**. The set $\mathcal{I}_S$ can be seen as the restriction of $\mathcal{I}_F = ([1 \cdots K] \times [1 \cdots l_s]) \cup ([1 \cdots K_s] \times \{l_s + 1\})$, definition set of a mapping independent of **b**, to $\mathcal{I}(\mathbf{b})$. Note that $|\mathcal{I}_F| = K_E$. On the sender side, the approach is thus straightforward and is illustrated in the example below. The decoder, knowing the values of the parameters l_s and K_s, can similarly compute the restriction of $\mathcal{I}_F$ to $\mathcal{I}(\widehat{\mathbf{b}})$ instead of $\mathcal{I}(\mathbf{b})$.

Example 2. Considering the source $\mathbf{S}_{(1)}$, the codes, and the sequence of symbols of Example 1, the SMA *algorithm* leads to the mapping of the stable indexes depicted in Figure 3. The notation $\varnothing$ stands for bitstream positions that have not been mapped during this stage. Remaining elements of the intermediate representation, here, respectively, 1 and 0001 for $\mathcal{C}_5$ and $\mathcal{C}_7$, are inserted in positions identified by the valuation $\varnothing$. This leads to the bitstreams **e** = 01101100 10001110 10 and **e** = 01111101 01101100 0111 for $\mathcal{C}_5$ and $\mathcal{C}_7$, respectively.

At that stage, some analogies with the first step of the EREC algorithm [9] can be shown. The EREC algorithm structures the bitstream in M slots, the goal being to create hard synchronization points at the beginning of each slot. The EREC algorithm thus leads to the creation of a constant mapping on a definition set $|\mathcal{I}_C| = Mh^- \le Kh^-$. Hence, it appears that for a number of slots lower than K (the number of symbols), the number of bits mapped in a constant manner is not maximized. This suggests using a constant mapping on the definition set $[1 \cdots K] \times [1 \cdots h_{\mathcal{C}}^-]$ and applying EREC on slots for the remaining bits to be mapped. The corresponding algorithm is called CMA-EREC in what follows. Note that if $M = K$, CMA-EREC is identical to EREC applied on a symbol basis (which satisfies $|\mathcal{I}_C| = Kh^-$). If $M = 1$, CMA-EREC is identical to CMA. The choice of M has a direct impact on the trade-off between resilience and complexity.

3.3. Stable mapping construction relying on a stack-based algorithm (SMA-stack)

This section describes an alternative to the above stable mapping algorithms offering the advantage of having a linear

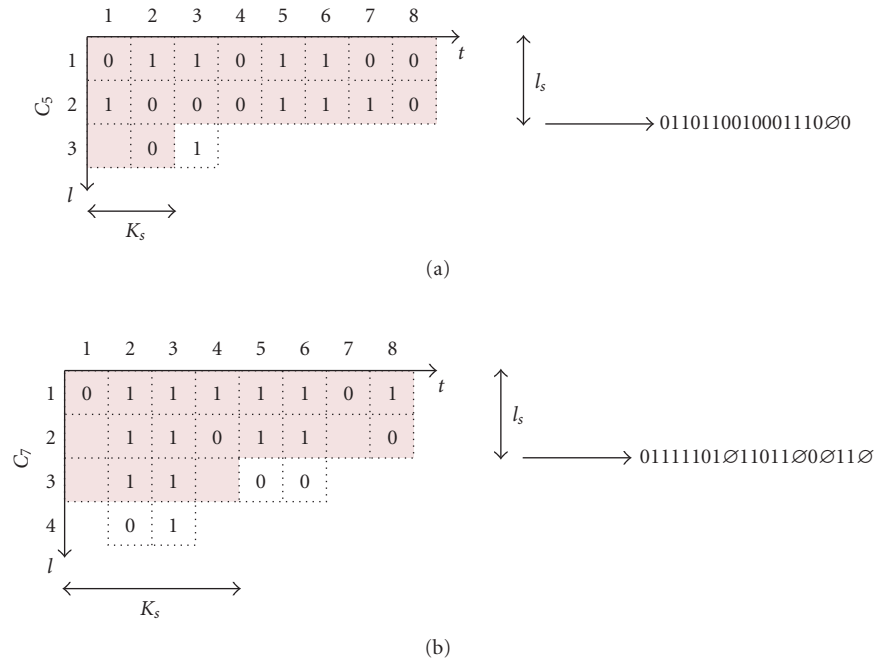

Figure 3: Definition set $\mathcal{I}_S$ (elements in gray) of the stable mapping (SMA algorithm).

complexity ($\mathcal{O}(K)$) with respect to EREC. Let us consider two stacks, Stack_b and Stack_p, dedicated to store bit values b_t^l of $\mathbf{b}$ and bitstream positions n of $\mathbf{e}$, respectively. These stacks are void when the algorithm starts. Let us consider a structure of the bitstream $\mathbf{e}$ in M slots with $M = K$, that is, with one slot per symbol s_t (the slots will be also indexed by t). The size of slot t is denoted m_t. There are K_s slots such that $m_t = l_s$ and $K - K_s$ slots such that $m_t = l_s + 1$. For each slot t, the algorithm proceeds as follows:

(1) the first $\min(m_t, L(s_t))$ bits of the codeword $\mathbf{b_t}$ associated to the symbol s_t are placed sequentially in slot t,
(2) if $L(s_t) > m_t$, the remaining bits of $\mathbf{b_t}$ (i.e., $b_t^{m_{t+1}} \cdots b_t^{L(s_t)}$) are put in the *reverse* order on the top of the stack Stack_b,
(3) otherwise, if $L(s_t) < m_t$, some positions of slot t remain unused. These positions are inserted on the top of the stack Stack_p,
(4) while both stacks are not void, the top bit of Stack_b is retrieved and inserted in the position of $\mathbf{e}$ indexed by the position that is on the top of the position stack Stack_p. Both stacks are updated.

After the last step, that is, once the slot K has been processed, both stacks are void. The decoder proceeds similarly by storing (resp., retrieving) bits in a stack Stack_b depending on the respective values of the codeword lengths $L(s_t)$ and of the slot size m_t. By construction of the slot structure, the number of stable elements is the same for both the SMA and the SMA-stack algorithm. The main difference between these algorithms resides in the way the remaining elements are mapped. Using the proposed stack-based procedure increases the error resilience of the corresponding bits.

Example 3. Let us consider again the source $\mathbf{S}_{(1)}$ of Example 1 with the code $\mathcal{C}_7$. Figure 4 illustrates how the SMA-stack algorithm proceeds. In this example, the slot structure has been chosen so that each slot is formed of contiguous bit positions, but this is not mandatory.

3.4. Layered bitstream

The previous *BC* algorithms decrease the impact of the error propagation induced by the channel errors. Another interesting feature is the amenability of the *BC* framework to progressive decoding. In Section 4.3, we will see that the different bit transitions of a binary codetree convey different amount of energy. In a context of progressive decoding, the bits which will lead to the highest decrease in reconstruction error should be transmitted first. This idea is underlying the principle of bit-plane coding in standard compression solutions. The approach considered here is however more general.

The approach consists in transmitting the bits according to a given order $\succ$. To each bit to be transmitted, one can associate an internal node of the codetree. One can thus relate the transmission order of the bits of the different codewords to a mapping of the internal nodes of the codetree into the bitstream. Thus, if $n_i \succ n_j$, all the bits corresponding to the internal node n_i are transmitted before any of the bits corresponding to the internal node n_j. This order induces a partition of the transitions on the codetree into segments of same "priorities." The bits corresponding to a segment of a given priority in the codetree are mapped sequentially. Note that this order may not be a total order: some transitions corresponding to distinct internal nodes may belong to the same

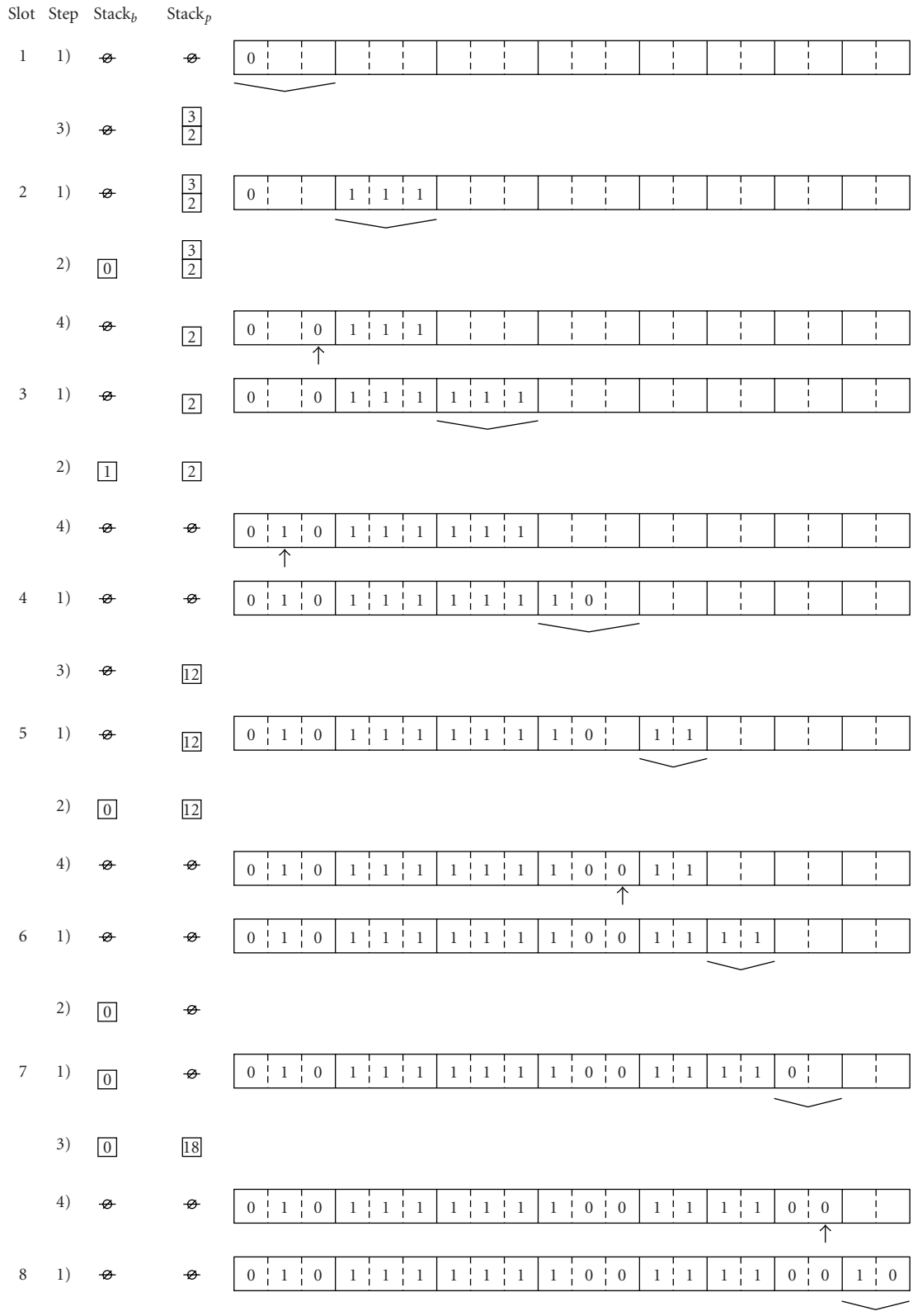

FIGURE 4: Example 3: encoding of sequence $a_1a_4a_5a_2a_3a_3a_1a_2$ using code $\mathcal{C}_7$ and SMA-stack algorithm.

segment. The order $\succ$ must satisfy the rule

$$n_j \not\succ n_i \quad \text{iff } n_j \in \mathcal{L}_i, \tag{7}$$

where $\mathcal{L}_i$ denotes the leaves attached to the node n_i in the binary codetree corresponding to the VLC code. This rule is required because of the causality relationship between nodes n_i and n_j.

Example 4. Let us consider again code $\mathcal{C}_5$. There are four internal nodes: the root /, 0, 1, and 10. These nodes are, respectively, referred to as n_0, n_1, n_2, n_3. A strict bit-plane[1] approach corresponds to the order $n_0 \succ n_1, n_2 \succ n_3$. Here, nodes n_1 and n_2 are mapped in the same segment. This order ensures that all the bits corresponding to a given "bit-plane" are transmitted before any of the bits corresponding to a deeper "bit-plane." Using this order, the realization $\mathbf{s} = a_1a_4a_5a_2a_3a_3a_1a_2$ of Example 1 is coded into the following bitstream:

$$\underbrace{01101100}_{n_0}\underbrace{1000110}_{n_1,n_2}\underbrace{01}_{n_3}. \tag{8}$$

Another possible order is the order $n_0 \succ n_2 \succ n_3 \succ n_1$. In that case, since the order is total, segments are composed of homogeneous bit transitions, that is, bit transitions corresponding to the same internal node in the codetree. Then, the realization $\mathbf{s}$ is transmitted as

$$\underbrace{01101100}_{n_0}\underbrace{0011}_{n_2}\underbrace{01}_{n_3}\underbrace{1010}_{n_1}. \tag{9}$$

Note that the CMA algorithm leads to construct a bitstream with a layered structure defined by the order $n_0 \succ n_1, n_2, n_3$. Note also that the concatenation of codewords corresponds to the less restrictive order between internal nodes, that is, for all (n_i, n_j), $n_i \not\succ n_j$.

The layered construction bitstream is an efficient way of enhancing the performance of unequal error protection (UEP) schemes. Most authors have considered the UEP problem from the channel rates point of view, that is, by finding the set of channel rates leading to the lowest overall distortion. For this purpose, RCPC [11] are generally used. The UEP problem is then regarded as an optimization problem taking as an input the relative source importance for the reconstruction. Here, the UEP problem is seen from the source point of view. It is summarized by the following question: how the bitstream construction impacts the localization of energy so that UEP methods may apply efficiently? Since the bit segments have a different impact on the reconstruction, these segments act as sources of various importance. In the usual framework, no distinction is performed among concatenated bits corresponding to different internal nodes. Using the layered approach, one can differentiate the bits according to their impact on the reconstruction and subsequently applies the appropriate protection. UEP techniques can thus be applied in a straightforward manner. The codetree itself has clearly an impact on the energy repartition, and has to be optimized in terms of the amount of source reconstruction energy conveyed by the different transitions on the codetree. The code design is discussed in Section 5.

4. DECODING ALGORITHMS

4.1. Applying the soft decoding principles (CMA algorithm)

Trellis-based soft decision decoding techniques making use of Bayesian estimators can be used to further improve the decoding SER and SNR performance. Assuming that the sequence § can be modeled as a Markov process, maximum a posteriori (MAP), maximum of posterior marginals[2] (MPM), or minimum of mean-square error (MMSE) estimators, using, for example, the BCJR algorithm [18], can be run on the trellis representation of the source model [19]. This section describes how the BCJR algorithm can be applied for decoding a bitstream resulting from a constant mapping (CMA algorithm).

Let us consider a symbol-clock trellis representation of the product model of the Markov source with the coder model [20]. For a given symbol index (or symbol clock instant) t, the state variable on the trellis is defined by the pair (S_t, N'_t), where N'_t denotes the number of bits *used to encode the first t symbols*. The value n'_t taken by the random variable N'_t is thus given by $n'_t = \sum_{t'=1}^{t} l(s_{t'})$. Notice that, in the case of classical transmission schemes where the codewords are simply concatenated, $n'_t = n$, where n is the current bit position in the bitstream $\mathbf{e}$.

The BCJR algorithm proceeds with the calculation of the probabilities $P(S_t = a_i \mid \hat{e}_1; \ldots; \hat{e}_{K_E})$, knowing the Markov source transitions probabilities $P(S_t = a_i \mid S_{t-1} = a_{i'})$, and the channel transition probabilities $P(\hat{E}_n = \hat{e}_n \mid E_n = e_n)$, assumed to follow a discrete memoryless channel (DMC) model. Using similar notations as in [18], the estimation proceeds with forward and backward recursive computations of the quantities

$$\alpha_t(a_i, n'_t) = P\big(S_t = a_i; N'_t = n'_t; (\hat{e}_{\varphi(t',l)})\big), \tag{10}$$

where $(\hat{e}_{\varphi(t',l)})$ denotes the sequence of received bits in bitstream positions $n = \varphi(t', l)$, with $1 \le t' \le t$ and $1 \le l \le L(s_{t'})$, and

$$\beta_t(a_i, n'_t) = P\big((\hat{e}_{\varphi(t',l)}) \mid S_t = a_i; N'_t = n'_t\big), \tag{11}$$

where $(\hat{e}_{\varphi(t',l)})$ denotes the sequence of received bits in bitstream positions $n = \varphi(t', l)$, with $t + 1 \le t' \le K$ and $1 \le l \le L(s_{t'})$. The recursive computation of the quantities

[1] The bit-plane approach for VLCs is somewhat similar to the one defined in the CABAC [16].

[2] Also referred to as symbol-MAP in the literature.

$\alpha_t(a_i, n'_t)$ and $\beta_t(a_i, n'_t)$ requires to calculate

$$\begin{aligned}\gamma_t(a_j, n'_{t-1}, a_i, n'_t) &= P\Big(S_t = a_i;\ N'_t = n'_t;\\ &\qquad (\widehat{e}_{\varphi(t,l)})_{1\le l\le L(a_i)} \mid S_{t-1} = a_j;\ N'_{t-1} = n'_{t-1}\Big)\\ &= \delta_{n_t - n_{t-1} - L(a_i)} P(S_t = a_i \mid S_{t-1} = a_j) \prod_{l=1}^{L(a_i)} P(\widehat{e}_{\varphi(t,l)} \mid b_t^l).\end{aligned} \tag{12}$$

In the case of a simple concatenation of codewords, $\varphi(t,l) = n'_{t-1} + l$. When using a constant mapping, we have

$$\varphi(t,l) = \begin{cases} (l-1)K + t & \text{if } l \le h^-_{\mathcal{C}},\\ (K-t)h^-_{\mathcal{C}} + n'_{t-1} + l & \text{otherwise.}\end{cases} \tag{13}$$

The product $\lambda_t(a_i, n') = \alpha_t(a_i, n')\beta_t(a_i, n')$ leads naturally to the posterior marginals $P(S_t, N'_t \mid \widehat{e}_1; \ldots; \widehat{e}_{K_E})$ and in turn to the MPM and MMSE estimates of the symbols S_t.

For the CMA algorithm, information on the bit and the symbol clock values is needed to compute the entities $\gamma_t(a_j, n'_{t-1}, a_i, n'_t)$. This condition is satisfied by the bit/symbol trellis [6]. However, this property is not satisfied by the trellis proposed in [21].

4.2. Turbo decoding

The soft decoding approach described above can be used in a joint source-channel turbo structure. For this purpose, extrinsic information must be computed on bits. This means computing the bit marginal probability bit $P(e_t = 0 \vee 1 \mid \widehat{e}_1; \ldots; \widehat{e}_{K_E})$ instead of the symbol marginal probability. The SISO VLC then acts as the inner code, the outer code being a recursive systematic convolutional code (RSCC). In the last iteration only, the symbol per symbol output distribution $P(S_t = a_i \mid \widehat{e}_1; \ldots; \widehat{e}_{K_E})$ is estimated.

4.3. MMSE progressive decoding

The above approach reduces the SER. However, it does not take into account MSE performance in a context of progressive decoding. Progressive decoding of VLC can be realized by considering an expectation-based approach as follows. Notice that VLC codewords can be decoded progressively by regarding the bit generated by the transitions at a given level of the codetree as a bit-plane or a layer.

Let us assume that the l first bits of a codeword have been received without error. They correspond to an internal node n_j of the codetree. Let $\mathcal{L}_j$ and $\widetilde{\mu}_j = \sum_{n_i \in \mathcal{L}_j} \mu_i$, respectively, denote the leaves obtained from n_j and the probability associated to the node n_j. Then the optimal (i.e., with minimum MSE) reconstruction value $\widetilde{a}_j$ is given by

$$\widetilde{a}_j = \frac{1}{\widetilde{\mu}_j} \sum_{n_i \in \mathcal{L}_j} \mu_i a_i. \tag{14}$$

The corresponding mean-square error (MSE), referred to as Δ_j, is given by the variance of the source knowing the first bits, that is, by

$$\Delta_j = \frac{1}{\widetilde{\mu}_j} \sum_{a_i \in \mathcal{L}_j} \mu_i (a_i - \widetilde{a}_j)^2. \tag{15}$$

Let us consider the codetree modeling the decoding process. The reception of one bit will trigger the transition from a parent node n_j to children nodes $n_{j'}$ and $n_{j''}$ depending on the bit realization. The corresponding reconstruction MSE is then decreased as $\Delta_j - \Delta_{j'}$ or $\Delta_j - \Delta_{j''}$ depending on the value of the bit received. Given a node n_j, the expectation δ_j of the MSE decrease for the corresponding transition T_j is given by

$$\delta_j = \Delta_j - \frac{\widetilde{\mu}_{j'}\Delta_{j'} + \widetilde{\mu}_{j''}\Delta_{j''}}{\widetilde{\mu}_{j'} + \widetilde{\mu}_{j''}}. \tag{16}$$

The term δ_j can be seen as an amount of signal *energy*. If all the bits are used for the reconstruction, the MSE equals 0, which leads to $\mathrm{var}(\mathbf{S}) = \Delta_{\mathrm{root}} = \sum_{n_j} \widetilde{\mu}_j \delta_j$, which can also be deduced from (16). The total amount δ_l^* of reconstruction *energy* corresponding to a given layer l of a VLC codetree can then be calculated as the weighted sum of *energies* given by transitions corresponding to the given layer:

$$\delta_l^* = \sum_{T_j \text{ in layer } l} \widetilde{\mu}_j \delta_j. \tag{17}$$

Remark 1. Note that the MMSE estimation can be further improved by applying a BCJR algorithm on the truncated bitstream, setting the transitions on the trellis that correspond to the nonreceived bits to their posterior marginals or to an approximated value of 1/2.

Note also that, if the quantities K and K_E are both known on the receiver side, error propagation can be detected if the termination constraints are not satisfied. Here, by termination constraints, we mean that the K_E bits of $\mathbf{E}$ must lead to the decoding of K symbols of $\mathbf{S}$. In the case where the termination constraint is not satisfied, it may be better to restrict the expectation-based decoding to the bits that cannot be de-synchronized (i.e., bits mapped with a constant or stable mapping).

5. CODE DESIGN

Based on our discussion in the previous section, the code should hence be optimized in order to satisfy at best the following criteria.

(1) In order to maximize the SNR performance in the presence of transmission errors, the code $\mathcal{C}$ should be such that it concentrates most of the *energy* on the bits (or codetree transitions) that will not suffer from de-synchronization. In particular, if the bits that concentrate most of the energy correspond to the first bit transition of the binary codetree, the concept of *most significant* bits can also apply for VLC codewords.

(2) Similarly, the progressivity depends on the amount of energy transported by the first transmitted bits. That is why the code design should be such that few bits

gather most of the reconstruction energy, and these bits should be transmitted first. For this purpose, we will assume that the layered approach proposed in Section 3.4 will be used.

(3) Finally, a better energy concentration enhances the performance of the UEP techniques: since these techniques exploit the fact that different sources (here segments of bits) have various priorities, the code should be designed to enhance the heterogeneity of the bit transitions in terms of reconstruction energy.

In this section, we will regard the code optimization problem as a simplified problem consisting in maximizing the values δ_l^* for the first codetree transitions. This problem can be addressed by optimizing (17), either with the binary switching algorithm [22] or with a simulated annealing algorithm (e.g., [23]). The optimization has to be processed jointly for every layer. Hence, this multicriteria optimization requires that some weights are provided to each layer l of (17). The weights associated to bit transition depend on the application. In the following, we propose two simplified approaches, led by the consideration that the lexicographic order separating the smaller values from the greater values—in general—concentrates most of the energy in the first layers. Obviously, a lexicographic order is relevant only for a scalar alphabet. The first approach consists in finding an optimal code (in the Huffman sense) that aims at satisfying a lexicographic order. The second approach consists in using Hu-Tucker [14] codes to enforce the lexicographic order.

5.1. Pseudolexicographic (p-lex) codes

Let us consider a classical VLC (e.g., using a Huffman code), associating a codeword of length l_i to each symbol a_i. One can reassign the different symbols a_i of the source alphabet to the VLC codewords in order to try to best satisfy the above criteria. In this part, the reassignment is performed under the constraint that the lengths of the codewords associated to the different symbols are not affected, in order to preserve the compression performance of the code. A new codetree, referred to as a *pseudolexicographic* (*p-lex*) VLC codetree, can be constructed as follows. Starting with the layer $l = h_c^+$, the nodes (including leaves) of depth l in the codetree are sorted according to their expectation value given in (14). Pairs of nodes are grouped according to the resulting order. The expectation values corresponding to the parent nodes (at depth $l-1$) are in turn computed. The procedure continues until the codetree is fully constructed. Grouping together nodes having close expectation values in general contributes to increase the energy or information carried on the first transitions on the codetree.

Example 5. Let us consider a Gaussian source of zeromean and standard deviation 1 uniformly quantized on 8 cells partitioning the interval $[-3,+3]$. The subsequent discrete source is referred to as $\mathbf{S}_{(2)}$ in what follows. Probabilities and reconstruction values associated to source $\mathbf{S}_{(2)}$ are given by $\mathcal{A} = \{-2.5112, -1.7914, -1.0738, -0.3578, 0.3578, 1.0738, 1.7914, 2.5112\}$ and $\boldsymbol{\mu} = \{0.01091, 0.05473, 0.16025, 0.27411, 0.27411, 0.16025, 0.05473, 0.01091\}$, respectively. The Huffman algorithm leads to the construction of the code $\{110100, 11011, 111, 01, 10, 00, 1100, 110101\}$ detailed in Table 2. The *p-lex* algorithm proceeds as in Table 1.

The reconstruction values obtained with this code are given in Table 2. Note that both the Huffman code and the code constructed with the *p-lex* algorithm have an edl of 2.521, while the source entropy is 2.471. The corresponding bit transition energies δ_j are also depicted. The reconstruction of symbols using the first bit only is improved by 1.57 dB (MSE is equal to 0.631 for the *p-lex* code instead of 0.906 for the Huffman code).

5.2. Hu-Tucker codes

For a given source, it may occur that the previous procedure leads to a code that preserves the lexicographic order in the binary domain. For example, it is well known that if the probability distribution function is a monotone function of symbol values, then it is possible to find a lexicographic code with the same compression efficiency as Huffman codes. But in general, it is not possible. In this section, Hu-Tucker [14] codes are used to enforce the lexicographic order to be preserved in the bit domain. The resulting codes may be suboptimal, with the edl falling into the interval $[h, h+2[$, where h denotes the entropy of the source. Thus, for the source $\mathbf{S}_{(2)}$, the edl of the corresponding Hu-Tucker code is 2.583, which corresponds to a penalty in terms of edl of 0.112 bit per symbol, against 0.050 for the Huffman code. The counterpart is that these codes have interesting progressivity features: the energy is concentrated on the first bit transitions (see Table 2). Thus, for the source $\mathbf{S}_{(2)}$, the reconstruction with the first bit only offers an improvement of 4.76 dB over Huffman codes.

6. SIMULATION RESULTS

The performance of the different codes and *BC* algorithms have been assessed in terms of SER, SNR, and Levenshtein distance with Source $\mathbf{S}_{(1)}$ and Source $\mathbf{S}_{(2)}$ (quantized Gaussian source), introduced in Examples 1 and 5, respectively. Let us recall that the Levenshtein distance [24] between two sequences is the minimum number of operations (e.g., symbol modifications, insertions, and deletions) required to transform one sequence into the other. Unless the number of simulations is explicitly specified, the results shown are averaged over 100 000 channel realizations and over sequences of 100 symbols. Since the source realizations are distinct, the number of emitted bits K_E is variable. Most of the algorithms that have been used to produce these simulation results are available on the web site [25].

6.1. Error resilience of algorithms CMA, SMA, and SMA-stack for p-lex Huffman and Hu-Tucker codes

Figures 5 and 6, respectively, show for Source $\mathbf{S}_{(1)}$ and Source $\mathbf{S}_{(2)}$ the SER and the Levenshtein distance obtained with the algorithms CMA, SMA, and SMA-stack in comparison with

Table 1: Construction of the code with the *p-lex* algorithm.

Bit transition level	nodes	selected aggregation	expectation	probability
6	$\{a_1, a_8\}$	$(a_1, a_8) \longrightarrow n_7$	$\mathbb{E}(n_7) = 0.000$	$P(n_7) = 0.022$
5	$\{n_7, a_2\}$	$(a_2, n_7) \longrightarrow n_6$	$\mathbb{E}(n_6) = -1.281$	$P(n_6) = 0.077$
4	$\{n_6, a_7\}$	$(n_6, a_7) \longrightarrow n_5$	$\mathbb{E}(n_5) = 0.000$	$P(n_5) = 0.131$
3	$\{n_5, a_3\}$	$(a_3, n_5) \longrightarrow n_4$	$\mathbb{E}(n_4) = -0.590$	$P(n_4) = 0.292$
2	$\{n_4, a_4, a_5, a_6\}$	$(n_4, a_4) \longrightarrow n_3$	$\mathbb{E}(n_3) = -0.478$	$P(n_3) = 0.566$
2	$\{a_5, a_6\}$	$(a_5, a_6) \longrightarrow n_2$	$\mathbb{E}(n_2) = +0.622$	$P(n_2) = 0.434$
1	$\{n_2, n_3\}$	$(n_2, n_3) \longrightarrow n_1$	$\mathbb{E}(n_1) = 0.000$	$P(n_1) = 1.000$

Table 2: Definition of Huffman, *p-lex* Huffman, and Hu-Tucker codes for a quantized Gaussian source. Leaves (e.g., alphabet symbols) are in italics.

Huffman node	$\mathbb{P}(n_j)$	$\widetilde{a}_j$	δ_j	*p-lex* node	$\mathbb{P}(n_j)$	$\widetilde{a}_j$	δ_j	Hu-Tucker node	$\mathbb{P}(n_j)$	$\widetilde{a}_j$	δ_j
$\varnothing$	1	0.000	0.022	$\varnothing$	1	0.000	0.297	$\varnothing$	1	0.000	0.626
0	0.434	+0.170	0.477	0	0.566	−0.4776	0.013	0	0.5	−0.791	0.228
1	0.566	−0.131	0.224	1	0.434	+0.6220	0.119	1	0.5	+0.791	0.228
00	0.160	+1.074	—	00	0.292	−0.5903	0.285	00	0.226	−1.317	0.145
01	0.274	−0.358	—	01	0.274	−0.358	—	01	0.274	−0.358	—
10	0.274	+0.358	—	10	0.274	+0.358	—	10	0.274	+0.358	—
11	0.292	−0.590	0.285	11	0.160	+1.074	—	11	0.226	+1.317	0.145
110	0.131	0.000	2.294	000	0.160	−1.074	—	000	0.066	−1.911	0.072
111	0.160	−1.074	—	001	0.131	0.000	2.294	001	0.160	−1.074	—
—	—	—	—	—	—	—	—	110	0.160	+1.074	—
—	—	—	—	—	—	—	—	111	0.066	+1.911	0.072
1100	0.055	+1.791	—	0010	0.077	−1.281	0.654	0000	0.011	−2.511	—
1101	0.077	−1.281	0.654	0011	0.055	+1.791	—	0001	0.055	−1.791	—
—	—	—	—	—	—	—	—	1110	0.055	+1.791	—
—	—	—	—	—	—	—	—	1111	0.011	+2.511	—
11010	0.022	0.000	6.306	00100	0.055	−1.791	—	—	—	—	—
11011	0.055	−1.791	—	00101	0.022	0.000	6.306	—	—	—	—
110100	0.011	−2.511	—	001010	0.011	−2.511	—	—	—	—	—
110101	0.011	+2.511	—	001011	0.011	+2.511	—	—	—	—	—

the concatenated scheme and a solution based on EREC [9] applied on a symbol (or codeword) basis, for channel error rates going from 10^{-4} to 10^{-1}. In Figure 5, the results in terms of SER and normalized distance have been obtained with code $\mathcal{C}_5$ (cf. Example 1). Figure 6 depicts the results obtained for a Huffman code optimized using the codetree optimization described in Section 5.1. This code is given in Table 2.

In both figures, it appears that the concatenated scheme can be advantageously replaced by the different *BC* algorithms described above. In particular, the SER performance of the SMA-stack algorithm approaches the one obtained with the EREC algorithm applied on a symbol basis (which itself already outperforms EREC applied on blocks of symbols) for a quite lower computational cost. Similarly, it can be noticed in Figure 7 that the best SNR values are obtained with Hu-Tucker codes used jointly with the EREC algorithm. It can also be noticed that the SMA-stack algorithm leads to very similar error-resilience performance. The results confirm that error propagation affects a smaller amount of reconstruction energy.

Remark 2. (i) For Source $\mathbf{S}_{(2)}$, Huffman and *p-lex* Huffman codes lead to the same error-resilience performance: the amount of energy conveyed by the bits mapped during the constant stage is identical for both codes. This can be observed in Table 2. However, for a large variety of sources, the *p-lex* Huffman codes lead to better results.

(ii) The layered bitstream construction has not been included in this comparison: the layered bitstream construction offers improved error resilience in a context of UEP. Simulation results depicted in Figures 5, 6, and 7 assume that no channel code has been used.

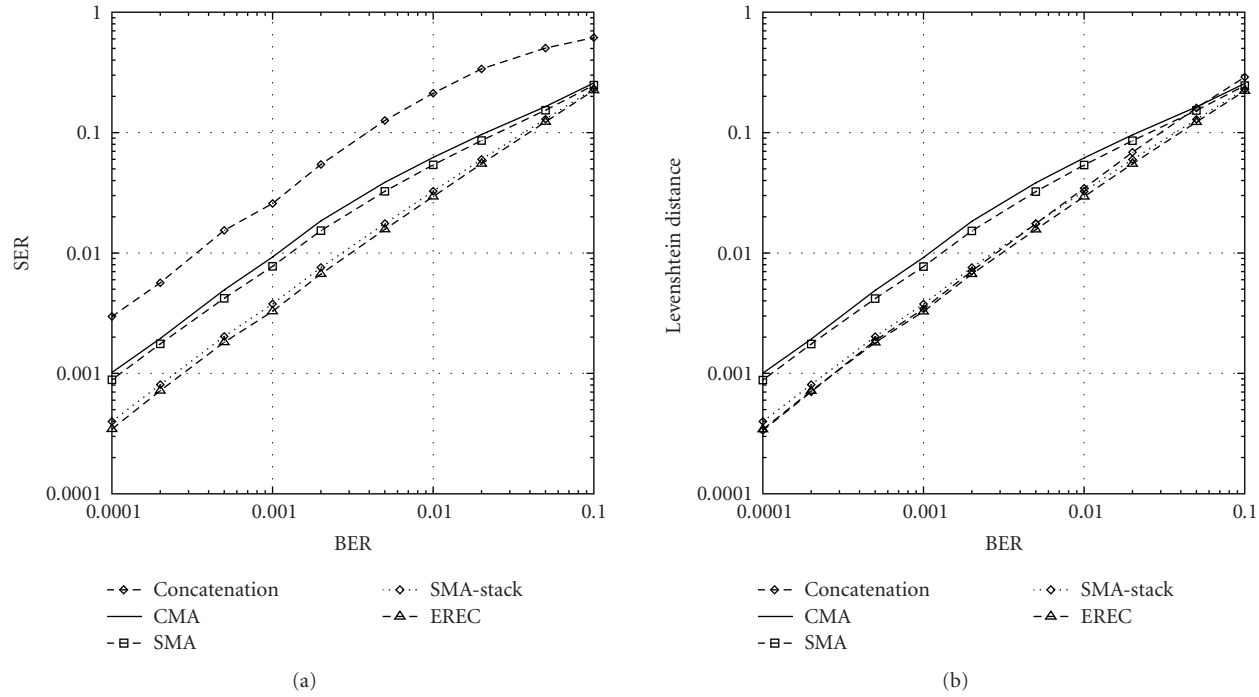

Figure 5: SER (a) and Levenshtein distance (b) performance of the different *BC* schemes for source $\mathbf{S}_{(1)}$ of Example 1 (code $\mathcal{C}_5$).

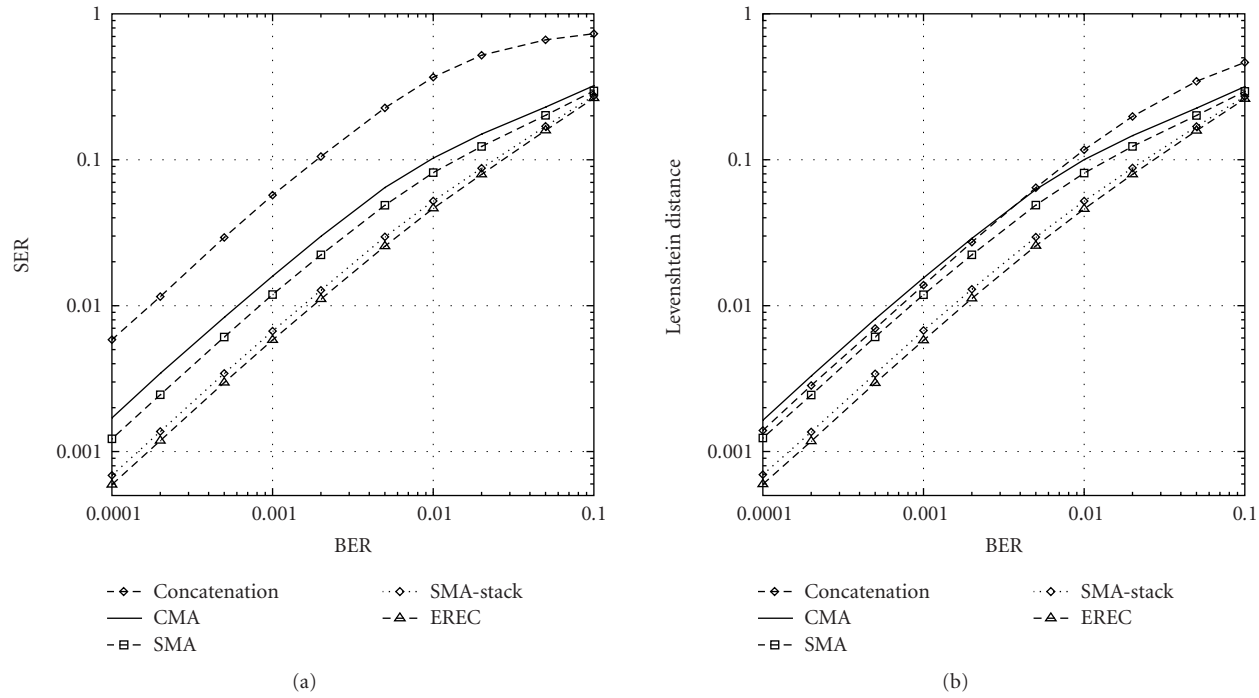

Figure 6: SER (a) and Levenshtein distance (b) performance of the different *BC* schemes with a quantized Gaussian source (source $\mathbf{S}_{(2)}$ encoded with Huffman codes).

6.2. *Progressivity performance of CMA and layered algorithms and impact of the code design*

The amenability to progressive decoding (using expectation-based decoding) of the CMA and layered solutions with Huffman, *p-lex*, and Hu-Tucker codes with respect to a concatenated scheme has also been assessed. These codes are also compared against lexicographic FLCs. For which, the first bit transitions notably gather most of the energy. For the layered approach, nodes have been sorted according to the value of

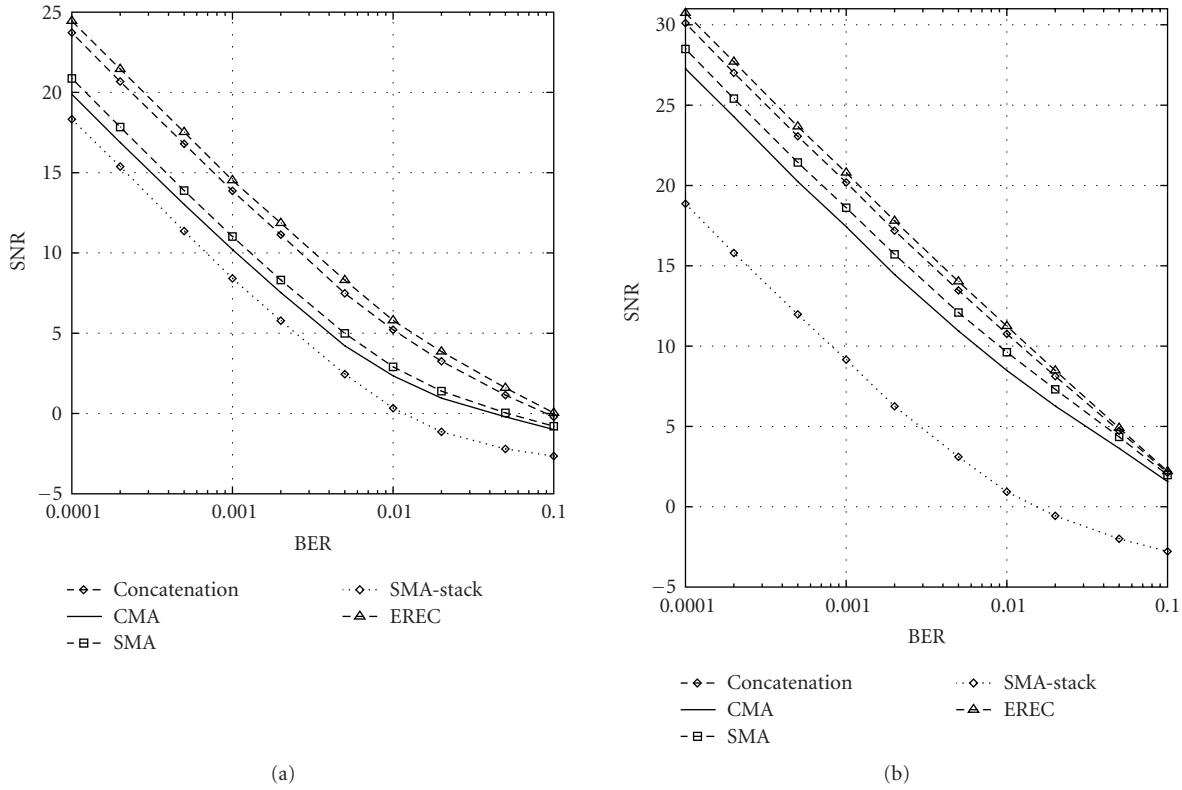

Figure 7: SNR performance of the different *BC* schemes (with pseudolexicographic Huffman) for (a) *p-lex* Huffman codes and (b) Hu-Tucker codes.

δ_j, under the constraint of (7). Let us recall that the values of δ_j are provided in Table 2. The corresponding orders between nodes (identified by the path in the codetree) are given hereafter. The corresponding values of δ_j are also given in Table 3.

The choice of the order has a major impact on the progressivity. Figure 8 shows the respective MSE in terms of the number of bits received for different approaches. The performance obtained is better than the one obtained with a bitplane FLC transmission, with, in addition, higher compression efficiency. The following points, identified on Figure 8 with a number, are of interest.

(1) The concatenation, which does not differentiate the bits, leads to the MSE performance that linearly decreases as the number of received bits increases.
(2) Several curves converge to this point. This is due to the fact that the number of bits received at this point corresponds to the fixed length portions of Huffman and *p-lex* Huffman codes, and that these codes have similar energy concentration properties on the next transitions on the codetree (see Table 2).
(3) For this source distribution, using the *p-lex* code instead of the usual Huffman code means transferring some *energy* from layer 2 (bits 101 to 200) to layer 1 (bits 1 to 100).
(4) For the Huffman and *p-lex* Huffman codes, the bit transitions corresponding to nodes with a high depth in the codetree bear a lot of reconstruction energy. As a result, the concatenation of codewords gives better performance than the layered transmission in general.

To conclude, this figure shows the efficiency of Hu-Tucker codes transmitted with a layered *BC* scheme with respect to classical VLCs and transmission schemes.

6.3. Error resilience with CMA

The performance in terms of SER of the CMA algorithm with an MPM decoder has been assessed against the one obtained with hard decoding and MPM decoding of concatenated codewords. The MPM decoder proceeds as described in Section 4.1. For this set of experiment, we have considered a quantized Gauss-Markov source with a correlation factor $\rho = 0.5$, and sequences of 100 symbols. Figure 9 shows the significant gain (SER divided by a factor close to 2) obtained with the CMA structure with respect to the concatenated bitstream structure for the same decoding complexity (the number of states in the trellis is strictly the same for both approaches).

6.4. Turbo decoding performance of CMA

In a last set of experiments, we have compared the amenability to turbo decoding (using expectation-based decoding) of the CMA solution with respect to a concatenated scheme,

Table 3: Ordering of the nodes for the layered approach and corresponding energies δ_j.

Huffman	∅	≻	0	≻	1	≻	11	≻	110	≻	1101	≻	11010
	0.022		0.477		0.224		0.285		2.294		0.654		6.306
p-lex Huffman	∅	≻	1	≻	0	≻	11	≻	110	≻	1101	≻	11010
	0.297		0.119		0.013		0.285		2.294		0.654		6.306
Hu-Tucker	∅	≻	0	≻	1	≻	00	≻	11	≻	000	≻	111
	0.626		0.228		0.228		0.145		0.145		0.072		0.072
FLC	∅	≻	0	≻	1	≻	01	≻	10	≻	00	≻	11
	0.626		0.190		0.190		0.119		0.119		0.072		0.072

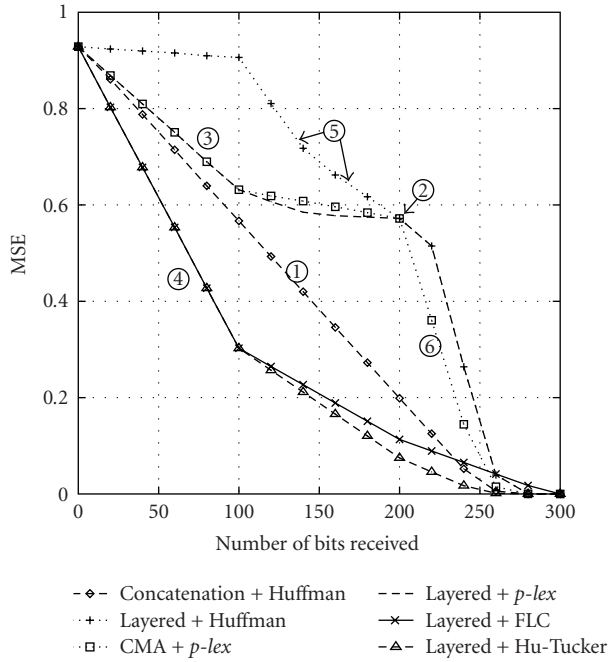

Figure 8: Progressive MSE performance/energy repartition profiles of the different codes and *BC* schemes without transmission noise.

both using Huffman codes. The decoding algorithm used for this simulation has been described in Sections 4.1 and 4.2. We have considered the first-order Markov source given in [7]. This source takes its value in an alphabet composed of three symbols. The matrix of transition probabilities is defined as

$$\mathbb{P}(S_t/S_{t-1}) = \begin{bmatrix} 0.94 & 0.18 & 0.18 \\ 0.03 & 0.712 & 0.108 \\ 0.03 & 0.108 & 0.712 \end{bmatrix}. \quad (18)$$

The channel considered here is an AWGN channel. Results have been averaged over 10000 realizations of sequences of length 100. Figure 10 shows the corresponding SER performance. This figure shows the clear advantage in terms of SER of the CMA algorithm. The improvement, expressed in terms of E_b/N_0, is around 0.4 dB. This gain is maintained as the number of iterations grows, and for the same complexity. For the Levenshtein distance measure, the concatenation leads to slightly better results. However, in multimedia applications, the SER is more critical than the Levenshtein distance.

7. DISCUSSION AND FUTURE WORK: ERROR-RESILIENT IMAGE CODING AND M-ARY SOURCE BINARIZATION

In this section, a simple wavelet image coder is described and is used to depict the interest of an error-resilient entropy coder for error-resilient encoding of real sources. The benefits and limitations of the method for M-ary source binarization in state-of-the-art coders are also discussed.

7.1. Image coder

The bitstream construction algorithms as well as the code design have been experimented with images. We have considered a simple image coding system composed of a 9/7

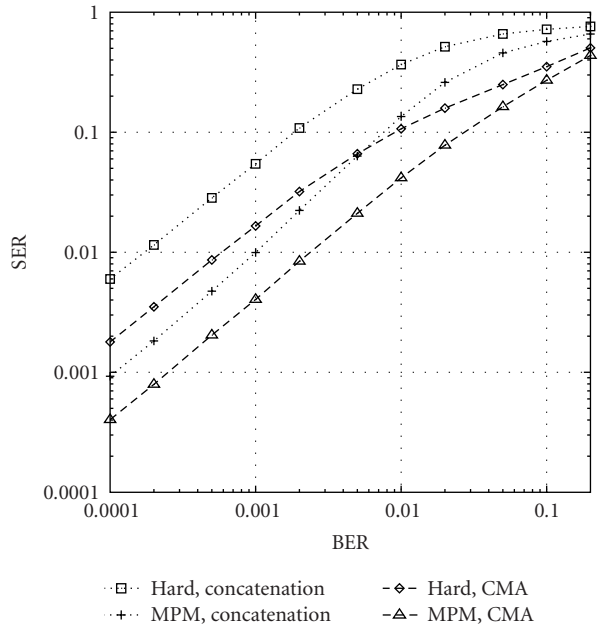

Figure 9: SER obtained by MPM decoding of a bitstream constructed by the CMA algorithm in comparison with those obtained with a concatenated bitstream structure (Gauss-Markov source with correlation $\rho = 0.5$).

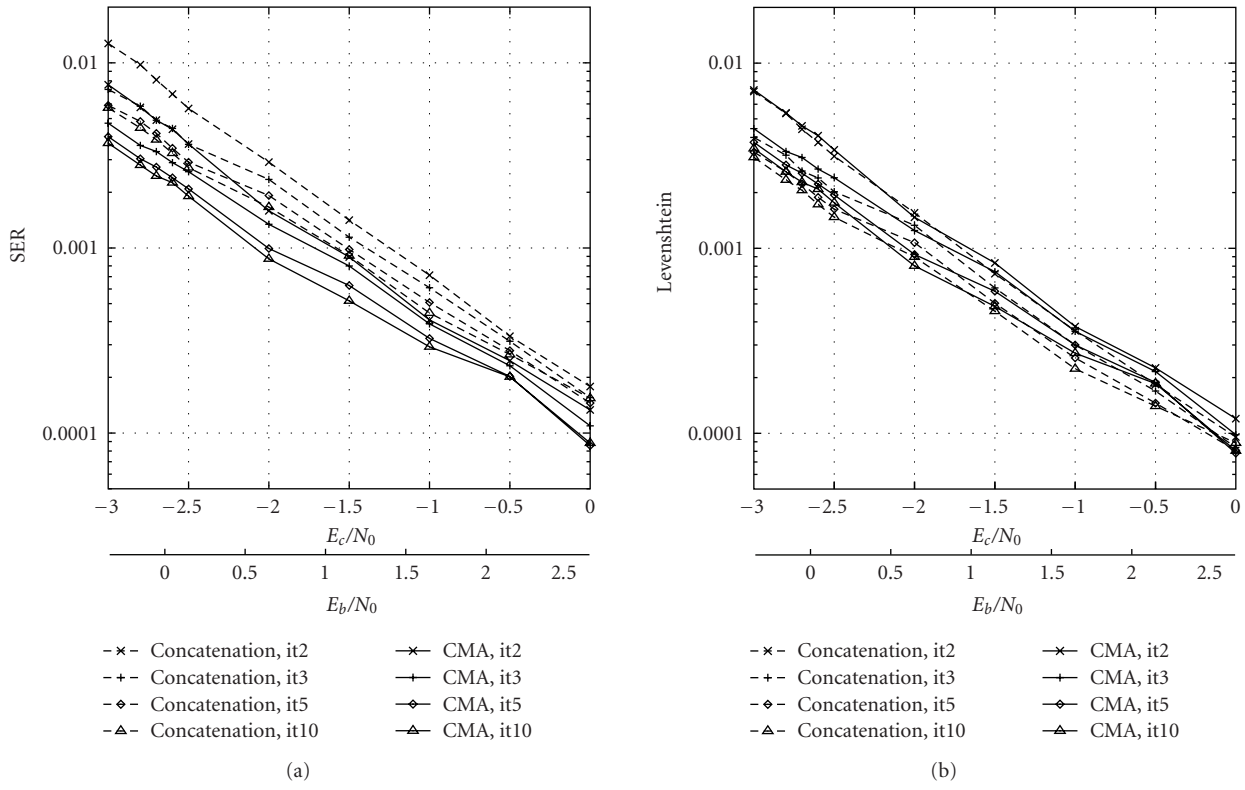

Figure 10: SER (a) and Levenshtein distance (b) obtained by two turbo decoding schemes, both of them composed of a source coder and an RSCC: (1) a classical joint-source channel turbo decoding, where the VLC codewords are concatenated; (2) a joint-source channel where the VLC codewords are encoded/decoded with the CMA bitstream construction algorithm.

Table 4: Parameters of the generalized Gaussian associated with the high frequency subbands (Lena).

Subband	*LLLLHL*	*LLLLLH*	*LLLLHH*	*LLHL*	*LLLH*	*LLHH*	*HL*	*LH*	*HH*
α	8.335	5.350	4.310	4.220	3.546	3.524	3.173	3.661	3.621
β	0.540	0.562	0.527	0.585	0.630	0.658	0.735	0.951	1.133
d_K	0.0355	0.0222	0.0210	0.0136	0.0102	0.0036	0.0018	0.0003	0.0001

Table 5: Error-resilience performance of *BC* algorithms.

BER	Huffman + concatenation	Huffman/Hu-Tucker + CMA	Huffman/Hu-Tucker + SMA-stack
10^{-4}	15.85	25.52	30.34
10^{-3}	11.16	22.25	25.76
10^{-2}	10.00	19.79	20.96

wavelet transform, a scalar uniform quantizer, and an entropy coder based on VLCs. The image is first decomposed into 10 subbands using a three-stage wavelet transform. Four scalar quantizers have been used, depending on the wavelet decomposition level to which the quantizer applies. The low subband has been uniformly quantized on 192 cells. The number of quantization cells for the high subbands is, respectively, given by 97, 49, 25, from the lowest to the highest frequencies. The corresponding quantization steps are equal to 10.56, 10.45, and 10.24, respectively. The pdf of high frequency subbands is then modeled as a generalized Gaussian of the form

$$p(x) = \frac{1}{c(\alpha,\beta)} e^{-|x|^{\beta}/\alpha}, \tag{19}$$

where $\alpha > 0$ and $\beta > 0$ are the parameters of the generalized Gaussian and where $c(\alpha,\beta)$ is a normalization constant. Note that the pdf is Gaussian if $\beta = 2$. To estimate the parameters of the generalized Gaussian distribution, we use a gradient algorithm with the Kullback-Leibler distance d_K. This distance measures the overhead induced by the model. The values of α, β obtained for the high frequency subbands of the image Lena are depicted in Table 4.

The obtained Kullback-Leibler distance values confirm that the generalized Gaussian model is a first-order accurate model of the distribution of image subbands. The pdf obtained from the parameter s can be synchronously computed on the encoder and on the decoder, assuming that the parameters are known on the decoder side. The probability laws are then used to design the VLC. For the low pass band, a Hu-Tucker code is chosen for the reasons exposed in Section 5. For this subband, it appears that the Hu-Tucker code has the same edl as the Huffman code. However, choosing this code for the high frequency dramatically increases the rate of the entropy coder, because the 0 symbol is assigned a codeword which is at least of length 2.

Hence, for the high frequencies, a Huffman code has been used. Our simple image coder gives a peak signal-to-noise ratio (PSNR) of 39.91 dB at a rate of 1.598 bit per symbol. These performance are obviously below the rate-distortion performance of state-of-the-art coders, but the comparative results in terms of error resilience are however of interest. Note that, unlike UEP schemes, an error resilient scheme such as the one proposed here does not require the knowledge of the channel characteristics. The gains achieved in terms of PSNR are given in Table 5. Each simulation point corresponds to the median value over 100 channel realizations for the image Lena. Figure 11 also depicts significant improvements in terms of visual quality for typical realizations of the channel.

7.2. *M*-ary source binarization

The entropy coders used in most state-of-the-art image and video coding systems rely on a first binarization step followed by bit-plane (or bin) encoding using, for example, arithmetic coding. This general principle is currently applied in EBCOT [15] and CABAC [16]. These algorithms will be all the most efficient both from a compression and error-resilience point of view if this binarization step makes use of a code which allows the concentration of most of the signal energy on the first transitions of the codetree of the binarization code. A priori analysis of the statistical distribution of the signal (transform coefficients, residue signals) to be encoded allows the design of a code with such properties. The scanning order of the bits resulting from the binarization step can then be defined as in Section 3 above. This scanning order minimizing the dependencies between bit planes will contribute to improve the error resilience of the entire coding approach.

If this binarization is coupled with an optimum entropy coder, the respective—entropy constrained—energy concentration properties of codes are modified. We noticed that in that particular case, the FLCs and the Hu-Tucker codes lead to similar rate-distortion curves. Moreover, the error sensitivity of the arithmetic coder to bit error suppresses the advantages of taking an error-resilient bitstream construction algorithm. Hence, the advantage of Hu-Tucker codes in this context should only be considered from a complexity point of view. However, coupled with the bitstream construction algorithm, they can be an efficient alternative to

Figure 11: PSNR performance and visual quality obtained, respectively, with concatenated Huffman codes and two transmission schemes using Hu-Tucker codes for low subband and Huffman codes for high subbands. These schemes use, respectively, the CMA algorithm and the SMA-stack algorithm. The channel bit error rates are 0.0001 (top images), 0.001 (middle images), and 0.01 (bottom images).

the Exp-Colomb or CAVLC recommended in H.264 for low-power mobile devices [16].

8. CONCLUSION

In this paper, we have introduced a bitstream construction framework for the transmission of VLC encoded sources over noisy channels. Several practical algorithms with different trade-offs in terms of error-resilience, progressivity, and feasibility of soft decoding have been described. The features of these algorithms are summarized in Table 6. Unlike the EREC algorithm, the complexity of the proposed algorithms increases linearly with the length of the sequence. For the CMA algorithm, the corresponding coding processes can be easily modeled under the form of stochastic automata which are then amenable for running MAP estimation and soft decision decoding techniques. Together with an MPM decoder, the CMA algorithm has been shown to increase the error-resiliency performance in terms of SER in comparison with a similar decoder for a concatenated bitstream, and this at no cost in terms of complexity. The approach has also been shown to offer improved SER performance in a turbo-VLC setup. For the other bitstream construction algorithms, that is, SMA, EREC, and SMA-stack, the design of efficient tractable soft and turbo decoding algorithms is a challenging problem. The code design has been shown to have a very high impact on the error resilience, the progressivity, and the amenability to enhance the UEP schemes. Hu-Tucker codes have been shown to be a good choice for these purposes. This framework can be applied to the problem of optimizing

Table 6: *BC* algorithms features.

BC algorithm	Linear complexity	Mandatory parameters	Soft decoding algorithms	Main purpose
Concatenation	YES	K or K_E	Bit/symbol trellis [6], bit-level trellis [21], ...	— —
CMA	YES	K	Bit/symbol trellis	Error resilience
SMA	YES	K and K_E	?	Error resilience
SMA-stack	YES	K and K_E	?	Error resilience
Layered	YES	K	?	Progressivity
EREC	NO	K and K_E	?	Error resilience

the source binarization step and in defining the appropriate scanning order of the bits resulting from the corresponding binarization step in modern image and video coders.

REFERENCES

[1] J. Maxted and J. Robinson, "Error recovery for variable length codes," *IEEE Transactions on Information Theory*, vol. 31, no. 6, pp. 794–801, 1985.

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

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

[4] Y. Takishima, M. Wada, and H. Murakami, "Reversible variable length codes," *IEEE Transactions on Communications*, vol. 43, no. 234, pp. 158–162, 1995.

[5] J. Wen and J. D. Villasenor, "Reversible variable length codes for efficient and robust image and video coding," in *Proceedings of Data Compression Conference (DCC '98)*, pp. 471–480, Snowbird, Utah, USA, March-April 1998.

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

[7] A. H. Murad and T. E. Fuja, "Joint source-channel decoding of variable-length encoded sources," in *Proceedings of IEEE Information Theory Workshop*, pp. 94–95, Killarney, Ireland, June 1998.

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

[9] D. W. Redmill and N. G. Kingsbury, "The EREC: an error-resilient technique for coding variable-length blocks of data," *IEEE Transactions on Image Processing*, vol. 5, no. 4, pp. 565–574, 1996.

[10] Y. Wang, S. Wenger, J. Wen, and A. K. Katsaggelos, "Error resilient video coding techniques," *IEEE Signal Processing Magazine*, vol. 17, no. 4, pp. 61–82, 2000.

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

[12] S. Cho and W. A. Pearlman, "Multilayered protection of embedded video bitstreams over binary symmetric and packet erasure channels," *Journal of Visual Communication and Image Representation*, vol. 16, no. 3, pp. 359–378, 2005.

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

[14] T. C. Hu and A. C. Tucker, "Optimal computer search trees and variable-length alphabetical codes," *SIAM Journal on Applied Mathematics*, vol. 21, no. 4, pp. 514–532, 1971.

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

[16] D. Marpe, H. Schwarz, and T. Wiegand, "Context-based adaptive binary arithmetic coding in the H.264/AVC video compression standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, no. 7, pp. 620–636, 2003.

[17] G. Zhou and Z. Zhang, "Synchronization recovery of variable-length codes," *IEEE Transactions on Information Theory*, vol. 48, no. 1, pp. 219–227, 2002.

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

[19] R. Bauer and J. Hagenauer, "Turbo-FEC/VLC-decoding and its application to text compression," in *Proceedings of the 34th Conference on Information Sciences and systems (CISS '00)*, pp. WA6–WA11, Princeton, NJ, USA, March 2000.

[20] A. Guyader, E. Fabre, C. Guillemot, and M. Robert, "Joint source-channel turbo decoding of entropy-coded sources," *IEEE Journal on Selected Areas in Communications*, vol. 19, no. 9, pp. 1680–1696, 2001.

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

[22] K. Zeger and A. Gersho, "Pseudo-Gray coding," *IEEE Transactions on Communications*, vol. 38, no. 12, pp. 2147–2158, 1990.

[23] N. Farvardin, "A study of vector quantization for noisy channels," *IEEE Transactions on Information Theory*, vol. 36, no. 4, pp. 799–809, 1990.

[24] V. I. Levenshtein, "Binary codes capable of correcting deletions, insertions and reversals," *Soviet Physics Doklady*, vol. 10, no. 8, pp. 707–710, 1966.

[25] H. Jégou, Source code: C++ implementation of the for proposed algorithms. http://www.irisa.fr/temics/Equipe/Jegou/src.

Hervé Jégou received the M.E. degree in industrial management from the Ecole Nationale Supérieures des Mines de Saint-Etienne in 2000 and the M.S. degree in computer science from the University of Rennes in 2002. From 2001 to 2003, he was also with the Ecole Normale Supérieure de Cachan. He is currently finishing his Ph.D. thesis at the University of Rennes. His research interests include joint-source channel coding and error-resilient source coding.

Christine Guillemot is currently "Directeur de Recherche" at INRIA, in charge of a research group dealing with image modelling and processing and video communication. She holds a Ph.D. degree from ENST (École 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 has 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 has served as an Associat Editor for the IEEE Transactions on Image Processing (2000–2003). She is currently serving as the Associat Editor for the IEEE Transactions on Circuits and Systems for Video Technology.