SUFFIX-CONSTRAINED CODES FOR PROGRESSIVE AND ROBUST DATA COMPRESSION: SELF-MULTIPLEXED CODES

Hervé Jégou and Christine Guilleminot

IRISA/INRIA, Campus universitaire de Beaulieu, 35042 Rennes

ABSTRACT
This paper addresses the issue of robust transmission of Variable Length Codes (VLCs) encoded sources over error-prone channels. A new class of codes, called self-multiplexed codes, is introduced. Their performance in terms of compression is the same as the one of classical VLCs (e.g., Huffman codes). Their proper use in energy distribution on respective transitions of the codetree allows to confine error propagation to transitions bearing low reconstruction energy. Simulation results reveal high performances in terms of signal to noise ratio.

1. INTRODUCTION
Entropy coding, producing variable length codewords, is a core component of any data 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 position 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]. Reversible VLCs [2] have also been designed to fight against de-synchronizations. Soft VLC decoding ideas, exploiting residual source redundancy (the so-called “excess-rate”) as well as the inter-symbol dependency, have also been shown to reduce the “de-synchronization” effect [3][4]. 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 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 as constraints by the decoding process. One can alternatively, by a proper structuring of the bitstream, reveal and exploit constraints on some bit positions. This idea is applied in [5] to blocks within an image. A structure of fixed length size slots inherently creates hard synchronization points in the bitstream. The resulting bitstream structure is called EREC (Error-Resilient Entropy Codes). Then, it appears that first transitions of VLC codetrees are mapped in a deterministic way.

The principle can however be pushed further. Thus, the design criterion that we consider is the amenability of VLCs and of transmission schemes for SNR error resilience, i.e., their signal to noise ratio (SNR) performance in presence of transmission errors. It is well known that codes with same compression efficiency may have different re-synchronizing properties. Similarly, codes with the same compression performances may have different energy concentration characteristics. Hence, VLC codetrees can be designed in such a way that most of the symbols energy is concentrated on transitions on the codetree corresponding to bits that will be mapped (on coder and decoder sides) in a deterministic way.

Given a classical VLC tree (e.g. a Huffman codetree), one can build a new codetree, by re-assigning symbols to leaves in order to satisfy the above criterion, leading to so-called pseudo-lexicographic codes. However, the amount of energy concentrated on the first layers of the tree strongly depends on the size and the structure of the alphabet. The codetree can be extended by assigning several fixed length codewords to a given symbol. A codeword can be seen as formed by a prefix representing the symbol followed by a suffix that will be used to represent subsequent symbols, hence the name self-multiplexed codes. These codes can also be regarded as multiplexed binary lexicographic [6] codes in the context of a unique source. The resulting compression properties of self-multiplexed codes are identical to those of classical VLCs. In contrast to pseudo-lexicographic codes, and similar to Fixed Length Codes (FLCs), the codeword assignment is not constrained by the tree structure. Hence, self-multiplexed codes provide higher degrees of freedom to optimize the signal energy concentration on the bits that will not suffer from any de-synchronization, leading in turn to higher SNR performances in presence of errors.

The rest of the paper is organized as follows. Section 2 introduces the problem addressed and the notations used in the sequel. Multiplexed codes are presented in section 3. A bit mapping algorithm is proposed in section 4. Expectation-based and progressive decoding properties are then considered in section 5 leading to the design of self-multiplexed codes (section 6). Simulation results are provided and discussed in section 7.

2. PROBLEM STATEMENT AND NOTATIONS
Let $S = (S_1, \ldots, S_K)$ be a sequence of source symbols taking their values in a finite alphabet $A$ composed of $\Omega$ symbols, $A = \{a_1, \ldots, a_\Omega\}$. In the sequel, we assume that $i_1 < i_2 \Rightarrow a_{i_1} < a_{i_2}$. Let $C$ be a binary variable length code designed for this alphabet, according to its stationary probability $\mu$. To each symbol $a_i$ is associated a codeword $C(a_i)$ denoted by $b_i = b_i^1 \ldots b_i^{l_i}$, where $l_i$ stands for the length of the codeword $b_i$. The term $l_i = L(S_i)$ represents the length of the codeword associated to the realization $s_i$ of the symbol $S_i$. In classical compression systems, the sequence of codewords produced would be transmitted sequentially, forming a concatenated bitstream. In the sequel, the emitted bitstream is denoted $E = E_1 \ldots E_K$, and its realization is denoted $e = e_1 \ldots e_K$. The length $K$ is assumed to be known at the time of decoding, but not necessarily $K_E$.

3. BINARY MULTIPLEXED CODES
A family of codes, called multiplexed codes, has been introduced in [7]. Their design principle relies on the fact that compression systems of real signals generate sources of information with different levels of priority (e.g., texture and motion information for a video signal). Two sources, a high priority source and a low priority source referred to respectively as $S_H$ and $S_L$, are considered. These codes are such
A binary multiplexed code $C^*$ can be constructed from a VLC code $C$ by choosing the length $l$ of the codewords of $C^*$ equal to the length $h_C^*$ of the longest codeword of $C$, i.e., $c = h_C^*$, and such that the cardinality $N_1$ of each equivalence class $C_1$ equals $2^{l_1}$, where $l_1$ is the length of the codeword describing $a_1$ in $C$. The amount of bits used to describe the symbol $a_1$ in $C^*$ is thus given by $c - \log_B(N_1) = l_1$. Therefore, the codes $C$ and $C^*$ have exactly the same compression properties. In the following, the length $l_1$ refers to the length $l_1 = c - \log_B(N_1)$ provided by the realization $s_i = a_i$ of $S_i$.

The next step in the construction procedure is the assignment of codewords to the different equivalence classes. Several assignment strategies can be considered.

We consider an index assignment according to the lexicographical order. We consider in addition the case where the way the bits $U_i$ index a particular codeword within an equivalence class is defined by the suffix of the codeword (the bits $U_i$ equal the codeword suffix). The code $C^*$ presented in Table 1 has been designed, using this construction procedure, from the Huffman code $C$ defined by $C(a_1) = 10$, $C(a_2) = 0$, $C(a_3) = 110$ and $C(a_4) = 111$.

These codes have been designed initially for robust transmission of two heterogeneous sources, that re-writing rules applied to lexicographical binary multiplexed codes lead naturally to the so-called self-multiplexed codes allowing for robust and progressive transmission of symbols of a unique source.

### 4. CONSTANT MAPPING ALGORITHM (CMA)

In this section, we describe a simple solution for constructing bitstreams. Given a code $C$, the solution is based on the mapping of the maximum number of bits into deterministic positions. The mapping positions of these bits do not depend on the source realization, hence is constant. The remaining bits to be mapped are then simply concatenated.

A variable length codetree comprises a section of a fixed length equal to the minimum length of a codeword denoted $h_C^*$, followed by a variable length section. Let $b'_l$ be a bit realization such that $l \leq h_C^*$. This bit can be allocated a bitstream position $\psi_C(t, l)$ as

$$(t, l) \mapsto \psi_C(t, l) = (l - 1)K + t$$

In contrast with classical transmission schemes where the codewords are concatenated, error propagation will only take place on the tuples $(t, l)$ such that $l > h_C^*$. This bit mapping amounts to 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 de-synchronization. Although, for classical VLCs, the CMA algorithm is outperformed by the EREC algorithm, we will see in section 7 that this mapping is efficient when used jointly with the new class of codes, called self-multiplexed codes, described in section 6. This layered bitstream structure is well suited for unequal error protection. Note also that this scheme is amenable to soft decoding using Bayesian estimation principles.

### 5. PROGRESSIVE DECODING AND SNR PERFORMANCE DESIGN CRITERIA

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. Progressive decoding can then be achieved by considering an expectation-based decoding approach.

#### 5.1 Expectation-based decoding

Let us assume that the $l$ first bits of a codeword have been received. They correspond to an internal node $n_i$ of the codetree. Let $C_i$ and $\bar{a}_j = \sum_{a_i \in \zeta_j} \mu_i a_i$ respectively denote the leaves deduced from $n_i$ and the probability associated to the node $n_i$. Then the optimal reconstruction value $\bar{a}_j$ is given by

$$\bar{a}_j = \frac{1}{\mu_j} \sum_{a_i \in \zeta_j} \mu_i a_i$$

The corresponding mean square error (MSE), referred to as $\Delta_j$, is given by the variance of the source knowing the first bits, i.e., by

$$\Delta_j = \frac{1}{\mu_j} \sum_{a_i \in \zeta_j} \mu_i (a_i - \bar{a}_j)^2.$$

Let us consider the codetree modelling 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 $\delta_j$ of the MSE decrease for the corresponding transition $T_j$ is given by

$$\delta_j = \Delta_j - \frac{\mu_{j'} \Delta_{j'} + \mu_{j''} \Delta_{j''}}{\mu_{j'} + \mu_{j''}}.$$

The total amount $\delta_T$ 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:

$$\delta_T = \sum_{T_j \in \text{layer } l} \mu_j \delta_j$$
5.2 Progressive SNR design criteria

Since the last bits of a codeword are more likely to be desynchronized, in order to maximize the SNR performance in the presence of transmission errors, the code is designed so that it concentrates most of the energy on the bits (or codetree transitions) corresponding to the first layers. The code design can be expressed as the maximization of the values $\delta_i$ for the bit transitions of the first layers. Since the first layers (which can be regarded as most significant bits) bring more information than subsequent layers, codes designed according to this criterion will lead to better SNR performances in a context of progressive decoding.

Note that, if the quantities $K$ and $K_E$ are both known, error propagation can be detected if the termination constraints are not verified. Here, by termination constraints, we mean that the $K_E$ bits of $\mathbf{E}$ must lead to decode $K$ symbols of $\mathbf{S}$. In that case, it may be better to restrict the expectation based decoding to the bits that cannot be desynchronized. This approach is referred to as truncated in the sequel, and offers its best results if the subset of bits used for the reconstruction contains a sufficient amount of energy.

5.3 Pseudo-lexicographic ($p$-lex) codes

Given a classical VLC, e.g., a Huffman code, which specifies the length $l_i$ of each symbol $a_i$, symbols can be re-assigned to leaves in order to best satisfy the above criterion. A new codetree can then be constructed using a bottom-up approach, leading to a so-called pseudo-lexicographic variable length codetree. The code construction can be summarized as follows. Starting with the bottom layer, referred to as the layer $h_2$, the nodes (including leaves) are sorted according to their expectation value given in Eqn. 2. Pairs of nodes are grouped according to the resulting order. The corresponding parent nodes are in turn assigned their reconstruction value. The processing of the codetree continues until the codetree is fully constructed. The code constructed with this method is referred to as $p$-lex in the sequel.

6. SELF-MULTIPLEXED CODES: DESIGN AND RELATED SCHEMES

The amount of energy concentrated on the first layer depends on the properties of the alphabet $A$. The alphabet, hence the codetree, can be extended by assigning several fixed length codewords to a given symbol. A codeword can then be seen as formed by a prefix representing the symbol followed by a suffix that will be used to represent subsequent symbols, hence the name self-multiplexed codes. In contrast with the pseudo-lexicographic codes, and similarly to FLCs, the codeword assignment, being not constrained by the tree structure, leads to the best concentration of signal energy, hence to a reduced reconstruction MSE for the first layers.

Self-multiplexed codes can thus be regarded as lexicographic binary multiplexed codes in the particular case of a unique source. In binary multiplexed codes, as explained in section 3, the codewords of a given equivalence class $C_i$ represent the same symbol $a_i$ of the high priority source. Instead of describing some data of a low priority source, the suffix is used to describe codewords of subsequent symbols.

6.1 Self-multiplexed encoding and decoding with a concatenated mapping

On the encoder side, the choice of a codeword representing a given symbol $S_i$ is constrained by the suffix, which must correspond to the first bits of the next codeword(s). The suffix length to be used is given by $c-l_i$. As a consequence, the sequence must be encoded backward (from the last to the first symbol). The encoding process consists in selecting the codeword describing the targeted symbol which verifies the suffix constraint. For the last symbol, a suffix constraint may not exist. In that case, the suffix constraint is arbitrarily assumed to be a sequence of $0$s, identified in the following example with zeros in italic.

Example 1: Let $A = \{a_1, a_2, a_3, a_4\}$ be the alphabet of the source $\mathbf{S}$ with the stationary probabilities given by $\mu_1 = 0.30, \mu_2 = 0.43, \mu_3 = 0.25$ and $\mu_4 = 0.02$. Let $s = a_1a_2a_3a_1a_2$ be a given realization of the source $\mathbf{S}$. With the code $C$ of table 1, the emitted bitstream $e = e_1e_2\ldots e_{k_E}$ is constructed as shown in table 2.

The sequence of bits received is decoded from the first bit to the last bit as follows: c bits of the bitstream are read, identifying the symbol $S_c$ to be decoded and its length $l_c$. A bitstream pointer (initialized on the first position) is then incremented by $l_c$ bitstream positions (and not c). Thus the last $c-l_c$ bits are re-used for decoding the following symbols.

6.2 Link with Rewriting Rules

Formally, the encoding of self-multiplexed codes using the concatenation of codewords can also be seen as a rewriting system of the sequence $\mathbf{S}$ using production rules (implicitly defined by the code) of the form

$$a_i \rightarrow \text{codeword},$$

For example, the code of table 1 amounts to define the following rewriting rules:

$$a_10 \rightarrow 000 \quad a_11 \rightarrow 001$$
$$a_20 \rightarrow 010 \quad a_211 \rightarrow 011$$
$$a_30 \rightarrow 100 \quad a_301 \rightarrow 101$$
$$a_3110 \rightarrow 111$$

Specific starting rules (for the last symbols of the sequences), are used if the last symbol is not sufficient to trigger a production rule by itself. They can be defined by assuming that missing bit(s) equal 0. Then, the production rules can be applied. The purpose of these rules is to transform the sequence $s$ into the sequence $e$ of bits. Any segment of the current sequence (composed of symbols and bits, initialized by $s$) can be rewritten if there exists a rule having this segment as an input (this input is composed of one symbol and a variable number of bits). When the production rules stop, the sequence contains only bit entities. On the decoder side, the decoding can be processed forward using reverse rules, e.g., $000 \rightarrow a_i0$. This formalism can be somewhat related to the grammar codes, recently analyzed in [8]. To the best of our knowledge, this kind of codes has never been studied in the context of error resilient source coding.

6.3 Self-multiplexed encoding and decoding with the CMA

In order to use the CMA algorithm jointly with self-multiplexed codes, the backward encoding procedure pre-
Table 3: Example 2: Self-multiplexed codes with CMA

<table>
<thead>
<tr>
<th>t</th>
<th>s</th>
<th>cWd</th>
<th>c1</th>
<th>c2</th>
<th>c3</th>
<th>c4</th>
<th>c5</th>
<th>c6</th>
<th>c7</th>
<th>c8</th>
<th>c9</th>
<th>c10</th>
<th>c11</th>
<th>c12</th>
<th>c13</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>a1</td>
<td>100</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>a1</td>
<td>000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>a1</td>
<td>000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>a2</td>
<td>100</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>a3</td>
<td>110</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>a2</td>
<td>010</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>a2</td>
<td>010</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>a1</td>
<td>001</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The decoding proceeds with first extracting $h_0^*$ bits from the bitstream constant section. The variable part of the codeword is processed by extracting $c - h_0^*$ bits from the bitstream variable segment. The suffix constraint length is given by $L_c - h_0^*$ bits. The suffix constraint is then re-used to decode the following symbols. Practically, a bitstream pointer, starting at the first position of the variable segment, is incremented by $L_c - h_0^*$ positions.

6.4 Localization of source energy

For all these codes, one can calculate the reconstruction energy $E_i$ provided by a given layer $t$, as explained in section 5. For example, for the Huffman code $C$ introduced in section 3, the reconstruction energy corresponding to the layers is given by $E_i = 0.0019$, $E_i = 15.2827$, and $E_i = 0.4630$. The corresponding self-multiplexed code $C$ of table 1 leads to the values $E_i = 8.70418$, $E_i = 6.5831$, and $E_i = 0.4630$. This result illustrates the amenability of self-multiplexed codes to concentrate most of the reconstruction energy on the bit transitions. Self-multiplexed codes by themselves (i.e., with classical concatenated mapping) do not lead a priori to improved error resilience. However, we will see in section 5 when used jointly with a constant bitstream mapping, these codes allow to guarantee, in presence of transmission errors, hard synchronization of most of the signal (see SNR performances in absence of transmission errors) and at the same time they offer best progressivity characteristics.

7. SIMULATION RESULTS

The performances of the different codes and bitstream construction algorithms have been evaluated in terms of SNR with a Gaussian source of zero-mean and standard deviation $\sigma = 1$, uniformly quantized on 8 cells partitioning the interval $[-3, +3]$. The results shown are averaged over 1000 channel and source realizations of 200 symbols. The channel is a binary symmetrical channel (BSC).

The set of experiments aimed at comparing the proposed scheme with the concatenated scheme and a solution based on EREC [5] applied on a symbol (or codeword) basis, for channel error rates going from $10^{-4}$ to $10^{-1}$. In presence of transmission errors, the SNR performances in presence of transmission errors (the truncated segment corresponds to the segment mapped with deterministic positions only) are shown in Fig.1. The truncation is applied only if the termination constraint is not verified as explained in section 5. It can be observed that, in presence of transmission errors, the self-multiplexed codes outperform significantly (in terms of SNR) the p-lex Huffman codes used with the EREC algorithm. The results confirm that error propagation affects a smaller amount of reconstruction energy.

8. CONCLUSION

In this paper, a new class of codes, called self-multiplexed codes, has been introduced. These codes can be regarded as a form of lexicographic binary multiplexed codes for a unique source. They have the same compression properties as classical VLCs and can be designed to reveal interesting energy distribution properties, allowing to map most of the reconstruction energy in deterministic bitstream positions. These codes turn out to offer very advantageous features: low cost coding and decoding processes, high compression efficiency (as good as Huffman codes), with high SNR performances in presence of transmission noise. They are in addition well suited for progressive decoding.

REFERENCES