Bit- and Trellis- Based Soft-Decision Sequential Decoding for Variable-Length Error-Correcting Codes

Yuh-Ming Huang and Chien-Feng Lo
Dept. of Computer Science and Information Engineering,
National Chi Nan University
Puli, Taiwan, R.O.C.
Email:ymhuang@csie.ncnu.edu.tw

Yunghsiang S. Han
Graduate Institute of Communication Engineering,
National Taipei University
Taipei County, Taiwan, R.O.C.
Email: yshan@mail.ntpu.edu.tw

***Abstract*—Variable-length error-correcting codes (VLECCs) have recently received extensive attention because they can provide both compression and error-correction capabilities simultaneously. The larger free distance a VLECC has, the more redundancy the VLECC suffers from. The redundancy can be used to combat the channel noise effectively by using the technique of joint source-channel decoding (JSCD). However, for larger VLECCs, the high computational decoding complexity has prevented these codes from implementation in practice. In this work, a new bit metric with low computational complexity is derived first, then based on a code trellis rather than on a code tree, we proposed a maximum *a posteriori* (MAP) bit-level soft-decision sequential decoding algorithm and its two approximations. Simulation results indicate that both approximations can provide nearly the same performance as the MAP scheme while exhibiting a significantly lower complexity.**

I. Introduction

Source and channel coding can be implemented separately without asymptotical performance loss according to the separate theory of Shannon [1]. However, the separate theory does hold only when both source code and channel code are optimal, where the source coder removes all redundancy and the channel coder corrects all errors. In practical system, there is often residual redundancy in the source encoder's output due to constraints on complexity and delay that may limit compression performance. Thus, this separate strategy does not necessary lead to a feasible solution with the best performance. Usually, the decoder might use the redundancy to protect the transmitted sequence over a noisy channel by acting as a statistical estimator of the transmitted sequence. Hence, joint source-channel coding (JSCC) has received a lot of attention in the literature recently.

Recent research in JSCC can be divided into two directions. One is to redesign the variable length codes (VLCs) [2]–[6], called the variable-length error-correcting codes (VLECCs), to make them more error-resilient, but increase compression loss. Such approaches indicate using VLECC alone can simultaneously provide the compression and error-correction capabilities. The other is to derive efficient decoding algorithms (regarded as one joint source/channel decoder) [2], [3], [6], [7] for the system formed by a VLECC alone. In this work, we focus on the latter.

Buttigieg and Farrell [2], Bystrom *et al.* [3] [5], Bau and Hagenauer [7] all performed the trellis-based MAP techniques in joint source/channel decoding. Based on a modified form of Viterbi [8] algorithm, Buttigieg and Farrell [2], Bystrom *et al.* [3] [5] adopted symbol-level trellis structures for MAP decoding on VLECCs. Using the BCJR [9] algorithm, Bauer and Hagenauer [7] presented a bit-level soft-in/soft-out decoder for VLECCs based on a bit-level trellis structure. For long source sequence, while all possible source information (for example, Markov source model) and side information (for example, the number of totally transmitted symbols) are take into consideration for the construction of the corresponding trellis, all above approaches have a common serious drawback that the decoding complexity will become quite expensive due to the enormous number of the trellis states.

In order to reduce the decoding complexity, low-complexity but sub-optimal symbol-level and tree-based sequential decoding algorithms were proposed in [6] [3]. The former is based on a ZJ (Stack) [10] [11] algorithm, whereas the latter is Fano [12] based. Recently, Huang *et al.* [13] have proposed symbol-level and trellis-based soft-decision priority-first decoding algorithms which outperform the existing methods [3] [6] for VLECCs.

In this work, based on the ideas of [13] and [14], we continue to present a bit-level and trellis-based MAP soft-decision sequential decoding algorithm (denoted as MAPSDA) and its two approximations. Simulation results show that the decoding complexity of our previously proposed symbol-level based decoding algorithms [13] can be further reduced a lot with no degradation on performance in terms of symbol error probability.

II. Bit- and Trellis- Based Soft-Decision Sequential Decoding

Let $\boldsymbol{S} = \{s_1, s_2, \ldots, s_K\}$ be a set of source symbols and $\boldsymbol{C}$ denote the corresponding binary VLC that has K codewords $\{\boldsymbol{c}_1, \boldsymbol{c}_2, \ldots, \boldsymbol{c}_K\}$ with respective probabilities $\{\Pr(\boldsymbol{c}_1), \Pr(\boldsymbol{c}_2), \ldots, \Pr(\boldsymbol{c}_k)\}$. The respective lengths of the codewords are given by $\{\ell_1, \ell_2, \ldots, \ell_K\}$. Let $\boldsymbol{c}_i = c_i^1 c_i^2 \ldots c_i^{\ell_i}$, $1 \leq i \leq K$. $\boldsymbol{C}$ can be represented in the form of a tree structure. An example is given in Fig. 1, where N_i, $0 \leq i \leq 2$, denotes the internal nodes and E_j, $1 \leq j \leq 4$, denotes the external nodes. The bit sequence of

each codeword $\boldsymbol{c}_i$ can be obtained by traversing the path from the root (internal) node N_0 to the leaf (external) node E_i. (N_0) was put beside E_j, it means that whenever any leaf node is reached, the traversing will be restarted from the root node. Here, we call N_is as the tree states and one of those states will be associated with each bit of any codeword.

Let I_i^j be the set of indices of all codewords with the first j bits being the same as the first j bits of codeword $\boldsymbol{c}_i$. According to [14], for a source of independent symbols, $\Pr(\boldsymbol{c}_i)$ can be rewritten as

$$\Pr(\boldsymbol{c}_i) = \Pr(c_i^1)\Pi_{j=2}^{\ell_i} \Pr(c_i^j|c_i^1, c_i^2, \ldots, c_i^{j-1}), \quad 1 \le i \le K, \tag{1}$$

where

$$\Pr(c_i^1) = \sum_{\ell \in I_i^1} \Pr(c_\ell), \tag{2}$$

and

$$\Pr(c_i^j|c_i^1, c_i^2, \ldots, c_i^{j-1}) = \frac{\sum_{\ell \in I_i^{j-1}|c_\ell^j = c_i^j} \Pr(c_\ell)}{\sum_{\ell \in I_i^{j-1}} \Pr(c_\ell)}. \tag{3}$$

As given in Fig. 1, for the code $\boldsymbol{C}_1$, we have

$$\begin{aligned}
\Pr(c_3^1) &= Pr(\boldsymbol{c}_2) + Pr(\boldsymbol{c}_3) + Pr(\boldsymbol{c}_4) \\
\Pr(c_3^2|c_3^1) &= \frac{Pr(\boldsymbol{c}_3) + Pr(\boldsymbol{c}_4)}{Pr(\boldsymbol{c}_2) + Pr(\boldsymbol{c}_3) + Pr(\boldsymbol{c}_4)} \\
\Pr(c_3^3|c_3^1, c_3^2) &= \frac{Pr(\boldsymbol{c}_3)}{Pr(\boldsymbol{c}_3) + Pr(\boldsymbol{c}_4)} \\
\Pr(\boldsymbol{c}_3) &= \Pr(c_3^1)\Pr(c_3^2|c_3^1)\Pr(c_3^3|c_3^1, c_3^2).
\end{aligned}$$

A bitstream $\boldsymbol{x}$ is a concatenation of L codewords, indexed by μ, from $\boldsymbol{C}$. Let $\boldsymbol{x} = (\boldsymbol{c}_{\mu(1)}, \boldsymbol{c}_{\mu(2)}, \ldots, \boldsymbol{c}_{\mu(L)})$, and $N = \sum_{j=1}^{L} \ell_{\mu(j)}$ be the length of the bitstream in bits. In this work, the bitstream is binary-phase-shift-keying (BPSK) modulated and then sent over an memoryless and additive white Gaussian noise (AWGN) channel without channel coding protection, and the received vector in bits $\boldsymbol{r} = (r_1, r_2, \ldots, r_N)$ denotes the set of the transmitted bits corrupted by the noise. We also assume that no bits are deleted by the channel, and that the bitstream length N is known at the receiver.

Each transmitted bitstream $\boldsymbol{x}$ of length N can be represented by a unique path through a bit-level trellis, where, for each node (l, s), l denotes a position (level) in the bitstream $\boldsymbol{x}$ and s denotes the tree state N_s associated with the l-th bit. Since N is known, nodes near the final level have only those branches corresponding to appropriate bits to terminate in node $(N, 0)$. Figure 2 shows a simple example of trellis of a bitstream of length N. Once a trellis is constructed, an MAP decoding can be applied to it whenever the MAP metric of each branch is specified.

Let W_N denotes the collection of all bitstreams of length N formed by sequences of codewords in $\boldsymbol{C}$. For MAP decoding, a bitstream $\hat{\boldsymbol{v}} \in W_N$ needs to be selected such that

$$\Pr(\hat{\boldsymbol{v}}|\boldsymbol{r}) \ge \Pr(\boldsymbol{v}|\boldsymbol{r}) \tag{4}$$

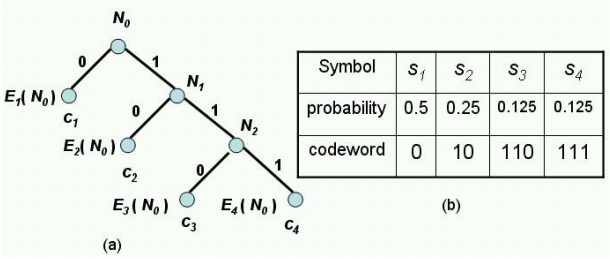

Fig. 1. Tree Structure for a code $\boldsymbol{C}_1$ with $K = 4$

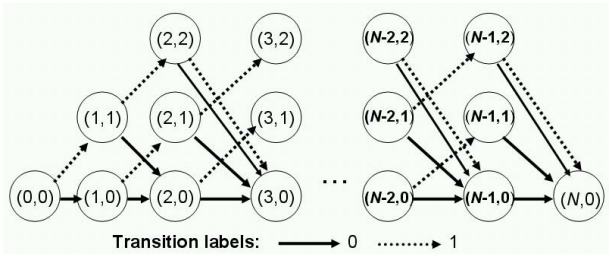

Fig. 2. Trellis diagram for the code $\boldsymbol{C}_1$

for all $\boldsymbol{v} \in W_N$. The hard-decision sequence $\boldsymbol{h} = (h_1, h_2, \ldots, h_N)$ corresponding to the received vector in bits $\boldsymbol{r} = (r_1, r_2, \ldots, r_N)$ is defined as

$$h_j \triangleq \begin{cases} 1, & \text{if } \phi_j < 0; \\ 0, & \text{otherwise,} \end{cases}$$

where

$$\phi_j \triangleq \ln \frac{\Pr(r_j|0)}{\Pr(r_j|1)}. \tag{5}$$

The condition in (4) can be rewritten as

$$\begin{aligned}
& \Pr(\boldsymbol{r}|\hat{\boldsymbol{v}})\Pr(\hat{\boldsymbol{v}}) \ge \Pr(\boldsymbol{r}|\boldsymbol{v})\Pr(\boldsymbol{v}) \\
\Leftrightarrow \quad & \sum_{j=1}^{N}(-1)^{\hat{v}_j}\phi_j + 2\ln\Pr(\hat{\boldsymbol{v}}) \ge \sum_{j=1}^{N}(-1)^{v_j}\phi_j \\
& +2\ln\Pr(\boldsymbol{v}) \qquad (6) \\
\Leftrightarrow \quad & \frac{1}{2}\sum_{j=1}^{N}[(-1)^{h_j} - (-1)^{\hat{v}_j}]\phi_j - \ln\Pr(\hat{\boldsymbol{v}}) \le \\
& \frac{1}{2}\sum_{j=1}^{N}[(-1)^{h_j} - (-1)^{v_j}]\phi_j - \ln\Pr(\boldsymbol{v}) \\
\Leftrightarrow \quad & \sum_{j=1}^{N}(h_j \oplus \hat{v}_j)|\phi_j| - \ln\Pr(\hat{\boldsymbol{v}}) \le \sum_{j=1}^{N}(h_j \oplus v_j)|\phi_j| \\
& - \ln\Pr(\boldsymbol{v}). \qquad (7)
\end{aligned}$$

The right side of equation (6) is the commonly adopted path metric for an MAP decoder such as the modified symbol-level Viterbi decoder proposed by Bystrom *et al.* [3]. However, Huang *et al.* [13] adopted the right side of equation (7) for proposing soft-decision symbol-level based priority-first decoding algorithms for VLECCs with better performance and lower computational complexity compared with results of [3] and [6]. In this work, we continue to propose soft-decision bit-level based priority-first decoding algorithms also based on equation (7).

Equation (7) indicates that the value of $\hat{\boldsymbol{v}}$ in W_N determines the path of length N ending at node $(N, 0)$ with the smallest metric in a trellis. Notably, $\hat{\boldsymbol{v}}$ may be not

the same as the transmitted bitstream $\boldsymbol{x}$. Moreover, since L is unknown to the decoder, even both bitstreams $\boldsymbol{x}$ and $\hat{\boldsymbol{v}}$ which respectively owns different number of codewords is possible.

Suppose $\boldsymbol{v} = (\boldsymbol{c}_{\tau(1)}, \boldsymbol{c}_{\tau(2)}, \ldots, \boldsymbol{c}_{\tau(M)})$, i.e., a concatenation of M codewords indexed by τ and $N = \sum_{m=1}^{M} \ell_{\tau(m)}$. Hence, under the assumption of independent source symbol, we have

$$-\ln \Pr(\boldsymbol{v}) = \sum_{m=1}^{M} -\ln \Pr(\boldsymbol{c}_{\tau(m)}). \tag{8}$$

Combined with equations (1) and (8), $-\ln \Pr(\boldsymbol{v})$ can be further rewritten as

$$-\ln \Pr(\boldsymbol{v}) = \sum_{m=1}^{M} \Big[-\ln \Pr(c^1_{\tau(m)}) + \sum_{n=2}^{\ell_{\tau(m)}} -\ln \Pr(c^n_{\tau(m)} | c^1_{\tau(m)}, c^2_{\tau(m)}, \ldots, c^{n-1}_{\tau(m)}) \Big]. \tag{9}$$

Thus, according to the right side of equation (7) and equation (9), the bit metric for each bit branch of the path associated with $\boldsymbol{v}$ in a trellis can be easily obtained. For example, the first bit metric is $(h_1 \oplus v_1)|\phi_1| - \ln \Pr(c^1_{\tau(1)})$.

Owing to the non-negativity of each bit metric, the metric value of the partial path is non-decreasing along any path in a trellis . Accordingly, the MAPSDA can be made to operate on a trellis, instead of on a code tree, as with the traditional stack algorithm. This is achieved by introducing a node table to record all nodes in the trellis that has been expanded, i.e., whose children have been visited. Based on the above definitions and notations, the MAPSDA can be presented.

⟨**The bit- and trellis- based MAPSDA for VLECCs** ⟩

Step 1. The Stack initially has only one original node (0,0) whose metric is assigned zero.

Step 2. If the top path in the Stack ends at the terminal node (N,0) in the trellis, then the algorithm stops.

Step 3. Delete the top path from the Stack, and put the ending node of this top path into the node table. Calculate the path metrics of the successors of the top path in the Stack.

Step 4. If any of the new paths ends at a node already in the node table, then discard the new path.

Step 5. If any of the new paths merges with a path already in the Stack, then remove the one with higher path metric value.

Step 6. Insert the remaining new paths into the stack and reorder the Stack according to ascending metric values. Go to Step 2.

As indicated in the previous algorithm, the *Stack* contains all paths explored by the MAPSDA, which are not the prefix of all other paths in the *Stack*. The *Stack* appears to function similarly to the stack in the traditional sequential decoding algorithm. The *node table* records the information of the ending nodes of the paths that had previously been the top paths of the *Stack*. The optimality of the MAPSDA can be proved by a similar argument given in [15].

TABLE I
HUFFMAN CODE $\boldsymbol{C}_2$ AND VLECC $\boldsymbol{C}_3$ RESPECTIVELY GIVEN IN TABLE II AND TABLE III OF [6] FOR THE 26-SYMBOL ENGLISH ALPHABET

Symbol	Probability	$\boldsymbol{C}_2$	$\boldsymbol{C}_3$
E	0.14878570	001	0111
T	0.09354149	110	10110
A	0.08833733	0000	11101
O	0.07245769	0100	01000
R	0.06872164	0110	110101
N	0.06498532	1000	001001
H	0.05831331	1010	000100
I	0.05644515	1110	0010100
S	0.05537763	0101	0001111
D	0.04376834	00010	1101100
L	0.04123298	10110	1000001
U	0.02762209	10010	10011011
P	0.02575393	11110	00001010
F	0.02455297	01111	10011100
M	0.02361889	10111	10000101
C	0.02081665	11111	100011110
W	0.01868161	000111	100010000
G	0.01521216	011100	1000100110
Y	0.01521216	100110	1000111010
B	0.01267680	011101	0000110100
V	0.01160928	100111	00001100010
K	0.00867360	0001100	00001100101
X	0.00146784	00011011	000011011100
J	0.00080064	000110101	00001101111100
Q	0.00080064	0001101001	0000110111111100
Z	0.00053416	0001101000	00001101111111110
FD		1	3
ACL		4.15572	6.269

In order to further lower the decoding complexity of MAPSDA, the effect of the source priori probability can be disregarded with almost no performance degradation, that is, the term $-lnPr(\boldsymbol{v})$ in (7) was not included. This approximate solution is denoted as AMAPSDA. Moreover, one heuristic policy was adopted on MAPSDA and AMAPSDA to ensure that the ending node of the next searched path was within a threshold to the ending node of the farthest visited path. The resultant algorithms are respectively denoted as MAPSDA@ and AMAPSDA@ depending on the selected threshold value @. Even though this policy slightly degrades the error rate performance of the proposed algorithms, it reduces the decoding complexity drastically. The larger threshold value we select, the less degradation in performance the algorithms will have.

III. SIMULATION RESULTS OVER AWGN CHANNEL

This section examines the computational effort and the performance of the MAPSDA and its approximations by simulations over the AWGN channel. The binary codeword is assumed to be antipodally transmitted. Hence, for the AWGN channel, the received vector is given by

$$r_j = (-1)^{v_j}\sqrt{\mathcal{E}} + \lambda_j,$$

for $0 \le j \le N-1$, where $\mathcal{E}$ denotes the signal energy per channel bit, and $\{\lambda_j\}_{j=0}^{N-1}$ denote independent noise samples of a white Gaussian process with single-sided noise power per hertz N_0. The signal-to-noise ratio (SNR) for the channel is therefore $\mathrm{SNR} \triangleq \mathcal{E}/N_0$.

For the AWGN channel, since $\phi_j = \frac{4\sqrt{\mathcal{E}}}{N_0} r_j$, h_j can be further simplified as

$$h_j \triangleq \begin{cases} 1, & \text{if } r_j < 0; \\ 0, & \text{otherwise.} \end{cases}$$

TABLE II
AVERAGE (AVE) AND MAXIMUM (MAX) NUMBER OF BRANCH METRIC COMPUTATIONS

SNR_b	3 dB		4 dB		5 dB		6 dB	
method	AVE	MAX	AVE	MAX	AVE	MAX	AVE	MAX
B-VA	632697	643164	632697	643164	632697	643164	632697	643164
S-VA [3]	162678	168202	162678	168202	162678	168202	162678	168202
S-SA200 [6]	32742	115452	27619	45316	26416	28121	26076	26488
B-SA200	25497	86375	15536	64474	11990	54875	10907	49278
B-SA400	33706	206852	17842	134295	12103	100265	10901	64626
MAPSDA20	177141	203418	85822	101977	44248	52727	24922	34120
AMAPSDA60	43604	69079	19121	34123	12662	16029	11053	12424

TABLE III
AVERAGE (AVE) AND MAXIMUM (MAX) STACK SIZE

SNR_b	3 dB		4 dB		5 dB		6 dB	
method	AVE	MAX	AVE	MAX	AVE	MAX	AVE	MAX
B-VA	89	89	89	89	89	89	89	89
S-VA [3]	17	17	17	17	17	17	17	17
S-SA200 [6]	200	200	200	200	200	200	200	200
B-SA200	200	200	200	200	200	200	200	200
B-SA400	400	400	400	400	400	400	400	400
MAPSDA20	251	329	181	295	110	196	62	211
AMAPSDA60	411	698	252	630	129	406	75	328

The VLECC C_3 given in Table I with average codeword length (ACL) in bits and free distance (FD) equal to 6.269 and 3, respectively, for 26-symbol English alphabet was considered in our simulation. For the VLECC, the code redundancy with respect to the Huffman code was considered. Hence, the following discussions adopt the SNR per information bit, i.e., SNR_b = SNR/R, where R =$4.15572/6.269$.

One thousand iterations were performed in our simulation. For each iteration, according to the probability of Table I (memoryless source assumption), 1000 source symbols were randomly generated and encoded into a bitstream of length N, which was transmitted over the AWGN channels with SNR_b in the range 3-6 dB. Decoding was performed under the assumption that the side information on the number of transmitted bits N is available to the decoder, but the number of transmitted symbols L is unknown.

As revealed in the algorithm, the computational efforts of the MAPSDA are determined not only by the numbers of branch metrics evaluated, but also by the cost of searching and reordering the stack elements. However, a balanced-tree data structure [15] can be adopted in the stack implementation, to ensure that the latter cost becomes of comparable order to the former one. Therefore, the branch metric computation of the MAPSDA should be considered as the key determinant of algorithmic complexity. The empirical investigation of the average decoding complexity and the performance in terms of symbol error rate (SER), which is equal to the Levenshtein distance [16] divided by the number of symbols in the transmitted message, is now considered.

Let the original symbol-level based modified Viterbi [3] and stack [6] algorithms be respectively abbreviated as S-VA and S-SA. S-SA200 denotes a stack of maximum size 200 was built in the S-SA. Based on a two-dimensional trellis (similar as the Fig. 2 of this paper) instead of a one-dimension one as shown in Fig. 1 of [5], a bit-level variant of the S-VA was implemented and abbreviated as B-VA.

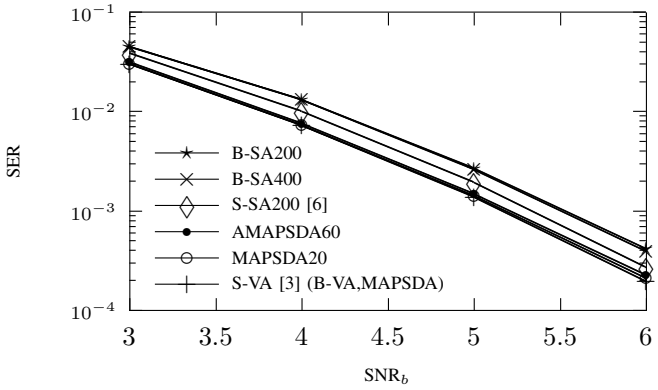

Fig. 3. Symbol error rate (SER) for 1000 randomly generated source symbols

Two bit-level variants of the S-SA with different stack size were also implemented and respectively abbreviated as B-SA200 and B-SA400. In Table II, we compare the average and maximum computational efforts of the proposed MAPSDA20 and AMAPSDA60 with those of the B-VA, S-VA [3], S-SA200 [6], B-SA200, and B-SA400. When $SNR_b \geq 5$ dB, the average number of branch metric (bit metric) evaluated in the B-SA200, B-SA400, or the AMAPPDA60 is reduced to about 12583 (=1000×6.269×2), which is near the smallest possible number of branch metric computations needed in any optimal bit-level based decoding algorithm. Notably, the computational effort of one branch metric (i.e. codeword metric) in the S-VA is about 6.269 times than that of one bit metric in the B-VA; the computational effort of one branch metric (i.e. codeword metric) in the S-SA is about 3.6345 (=(6.269+1)/(1+1)) times than that of one bit metric in the B-SA. Moreover, the bit metric computation in MAPSDA20 or AMAPSDA60 is far simpler than traditional bit metric computation performed in B-VA or B-SA.

In Table III, the average and maximum stack size required in each variety of decoding algorithms were also listed for fair comparison. Note that the cost of searching and reordering the stack elements is in logarithmicly proportion to the stack size. It can be seen that the proposed algorithms have smaller average stack sizes compared with stack algorithms when $SNR_b \geq 5$ dB.

In Fig. 3, at all SNR_b levels, both MAPSDA20 and AMAPSDA60 all have better performance than the S-SA200, B-SA400, and B-SA200, and achieve nearly the same performance on SER as the optimal decoders, B-VA and S-VA.

IV. Conclusions

In this work, we proposed an MAP decoding algorithm for variable-length error-correcting codes. This algorithm operates on bit-based trellises and posses a lower decoding complexity when compared with those existing algorithms operated on symbol-based trellises. The approximations of the proposed algorithm can further reduce the decoding complexity without significant error rate performance degradation. In the future, the performance of the proposed algorithm will be evaluated on a system where channel codes are implemented.

References

[1] C. E. Shannon, "A mathematical theory of communication," *Bell Sys. Tech. J.*, vol. 27, pp. 379–423, Jul. 1948.

[2] V. Buttigieg and P. G. Farrell, "A maximum likeligood decoding algorithm for variable-length error-correcting codes," in *Proc.* 5^{th} *Bangor Symp. on Communications*, Jun. 1993, pp. 56–59.

[3] M. Bystrom, S. Kaiser, and A. Kopansky, "Soft source decoding with applications," *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 11, no. 10, pp. 1108–1120, Oct. 2001.

[4] K. Lakovic and J. Villasenor, "On design of error-correcting reversible variable length codes," *IEEE Commun. Letters*, vol. 6, no. 8, pp. 337–339, Jan. 2002.

[5] A. Kopansky and M. Bystrom, "Extending soruce codes for error resilience," in *Proc. Data Compression Conf.*, Mar. 2003, pp. 23–32.

[6] V. Buttigieg and R. Deguara, "Using variable-length error-correcting codes in MPEG-4 video," in *Proc. Int. Symp. on Inform. Theory*, Sep. 2005, pp. 2379–2383.

[7] R. Bauer and J. Hagenauer, "On variable length codes for iterative source/channel decoding," in *Proc. Data Compression Conf.*, 2001, pp. 273–282.

[8] A. J. Viterbi, "Error bound for convolutional codes and an asymptotically optimum decoding algorithm," *IEEE Trans. Inf. Theory*, vol. IT-13, no. 2, pp. 260–269, Apr. 1967.

[9] L. R. Bahl, J. Cocke, F. Jelinek, and J. Raviv, "Optimal decoding of linear codes for minimizing symbol error rate," *IEEE Trans. Inf. Theory*, pp. 284–287, Mar. 1974.

[10] K. S. Zigangirov, "Some sequential decoding procedures," *Probl. Peredachi Inf., 2*, pp. 13–25, 1966.

[11] F. Jelinek, "A fast sequential decoding algorithm using a stack," *IBM J. Res. and Dev., 13*, pp. 675–685, Nov. 1969.

[12] R. M. Fano, "A heuristic discussion of probabilistic decoding," *IEEE Trans. Inf. Theory*, vol. IT-9, no. 2, pp. 64–73, Apr. 1963.

[13] Y.-M. Huang, Y. S. Han, and T.-Y. Wu, "Soft-decision priority-first decoding algorithms for variable-length error-correcting codes," *to appear in IEEE Communications Letters.*, 2008.

[14] 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, Jan. 2005.

[15] Y. S. Han, P.-N. Chen, and H.-B. Wu, "A maximum-likelihood soft-decision sequential decoding algorithm for binary convolutional codes," *IEEE Trans. Commun.*, vol. 50, no. 2, pp. 173–178, Feb. 2002.

[16] T. Okuda, E. Tanaka, and T. Kasai, "A method for the correction of garbled words based on the Levevshtein metric," *IEEE Trans. Comput.*, vol. C-25, pp. 172–178, Feb. 1976.