Transactions Letters

Soft-Input Soft-Output Decoding of Variable Length Codes

Jiangtao Wen and John Villasenor

***Abstract*—We present a method for utilizing soft information in decoding of variable length codes (VLCs). When compared with traditional VLC decoding, which is performed using "hard" input bits and a state machine, the soft-input VLC decoding offers improved performance in terms of packet and symbol error rates. Soft-input VLC decoding is free from the risk, encountered in hard decision VLC decoders in noisy environments, of terminating the decoding in an unsynchronized state, and it offers the possibility to exploit *a priori* knowledge, if available, of the number of symbols contained in the packet.**

***Index Terms*—Soft decoding, variable length codes.**

I. Introduction

In most applications of variable length codes (VLCs), decoding is performed bit by bit, with the input to the entropy decoder assumed to be a sequence of "hard" bits about which no soft information is available. However, in noisy environments, soft information can be associated with each information bit, either by direct use of channel observations in the case of uncoded transmission, or through soft-output channel decoders when channel coding is used. It is intuitive that this soft information, if it can be exploited, can be used to improve the performance of VLC decoding.

In recent years, various algorithms [1]–[6] have been proposed to explore the possibility of using soft information for VLC decoding. These algorithms improved decoding performance (as compared to traditional hard-bit-state-machine-based decoding) by optimizing decisions on symbols or packets instead of bits. Such optimization is done by utilizing source symbol distribution information and channel characteristics. In [1], a forward–backward estimation of source distribution was proposed. In [5] and [6], the authors explored the advantage of using soft decoding on reversible variable length codes (RVLCs). The redundancy in RVLCs, as expressed by reversibility, can be implicitly exploited by the soft decoder.

The main contribution of the present paper is the introduction of a soft-in soft-out (SISO) dynamic decoding algorithm for decoding of VLCs. The SISO approach is inspired by SISO channel decoding algorithms such as the soft-output Viterbi algorithm (SOVA) [7] and methods for decoding of turbo codes [8], although the application to VLCs is quite different in concept and implementation. The SISO VLC decoder receives as input a packet of known length containing VLC data that has been corrupted by additive white Gaussian noise (AWGN) and produces the codeword sequence which is most likely to have been input to the VLC encoder at the transmitter. Simulation results show significant improvement in performance relative to traditional hard decision decoding of VLCs. The SISO method performs best when *a priori* information regarding the number of symbols represented in the packet is available, though even in the case that the number of symbols is unknown, the performance is superior to hard decision methods.

Paper approved by M. Fossorier, the Editor for Coding and Communication Theory of the IEEE Communications Society. Manuscript received October 20, 1998; revised April 4, 2001, and September 24, 2001. This paper was presented in part at the 1999 IEEE Data Compression Conference, Snowbird, UT.

J. Wen is with the PacketVideo Corporation, San Diego, CA 92121 USA (e-mail: gwen@pv.com).

J. Villasenor is with the Electrical Engineering Department, University of California, Los Angeles, Los Angeles, CA 90024-1594 USA (e-mail: villa@icsl.ucla.edu).

Publisher Item Identifier S 0090-6778(02)05116-4.

II. SISO Algorithm Description

We consider a source producing symbols selected from an alphabet $\mathbf{X} = \{X_1, X_2, \ldots X_K\}$, where symbol X_k occurs with probability p_k and is represented using a binary codeword x_k of l_k bits. The alphabet size K may be infinite. The output of the VLC encoder is transmitted using packets of length L bits, representing N symbols. We use the integer $c(i)$ to denote the index to the symbol in the ith position in the packet; for example, if $c(i) = j$, then the symbol in the ith position is X_j. The vector of symbol indices $\boldsymbol{C} = \{c(1), c(2), \ldots c(N)\}$ is mapped by the VLC encoding to the binary sequence represented by the concatenation $\{x_{c(1)}x_{c(2)}\cdots x_{c(N)}\}$.

The information available at the input to the VLC decoder is the observation vector containing L noise-corrupted bits $\mathbf{O} = \{o_1, o_2, \ldots, o_j, \cdots, o_L\}$ that can be divided into N subvectors $\boldsymbol{O}_i\}_{i=1}^{N}$, each containing the $l_{c(i)}$ bits representing the symbol $X_{c(i)}$, so that $\mathbf{O}$ can also be written as the concatenation $\mathbf{O} = \{\mathbf{O}_1\mathbf{O}_2\cdots\mathbf{O}_N\}$. Based on the observation vector $\boldsymbol{O}$, the optimal decoder selects the symbol vector $\boldsymbol{C}$ that maximizes $\mathrm{Prob}\{\boldsymbol{C}|\boldsymbol{O}\}$, the posterior probability of the symbol sequence given the observation. This is equivalent to maximizing $\mathrm{Prob}\{\boldsymbol{C}, \boldsymbol{O}\}$, the joint probability of the symbol sequence and the observation. The vector that maximizes $\mathrm{Prob}\{\mathbf{C}, \mathbf{O}\}$ is denoted as $\mathbf{C}^* = \{c^*(1), c^*(2), \ldots c^*(N)\}$, and is associated with probability $P^* = \mathrm{Prob}\{\mathbf{C}^*, \mathbf{O}\} = \max_{\mathbf{C}} \mathrm{Prob}\{\mathbf{C}, \mathbf{O}\}$. For a packet of L bits and N symbols, $\boldsymbol{C}^*$ and P^* are calculated from Bellman's principle of dynamic programming [9], which leads to recursive equations as follows:

$$\begin{aligned}\boldsymbol{C}^*(N, L) &= \boldsymbol{C}^*\left(N-1, L-l_{c^*(N)}\right) \cup c^*(N)\\ P^*(N, L) &= P^*\left(N-1, L-l_{c^*(N)}\right)\\ &\quad \cdot \mathrm{Prob}\left\{o_{L-l_{c^*(N)}+1}, \ldots, o_L | c(N) = c^*(N)\right\}\\ &\quad \cdot p_{c^*(N)} \qquad (1)\end{aligned}$$

with

$$c^*(N) = \arg \max_{i,\, i\in[1,\,K]} \{P^*(N-1,\, L-l_i) \cdot \mathrm{Prob}\{o_{L-l_i+1}, o_{L-l_i+2}, \ldots, o_L | c(N)=i\} \cdot p_i\} \quad (2)$$

$$C^*(1,\, L) = \arg \max_{i,\, i\in[1,\,K]} \mathrm{Prob}\{o_1, o_2, \ldots, o_L | c(1)=i\} \cdot p_i. \quad (3)$$

The principal underlying (1) is that if the optimal last codeword in the packet is a codeword of length $l_{c^*(N)}$, the rest of the optimal packet, corresponding to $N-1$ symbols, must be the optimal $N-1$ symbol packet corresponding to the first $L-l_{c^*(N)}$ received values. Equations (2) and (3) involve selecting from the alphabet the symbol that is most likely, as measured by the products of probabilities that constitute the argument for the max function. The "SISO" terminology is used to describe the algorithm because the decoder provides not only the optimal sequence $\boldsymbol{C}^*$ but also the probability, or confidence level P^* associated with that sequence.

In the above algorithm, we assume that the decoder has *a priori* knowledge of L and N, i.e., the length of the packet and the number of symbols contained. This is a valid assumption in many applications and practical systems. For example, when compressed video content is transmitted over bit error prone networks using the MPEG-4 error resilient syntax with data partitioning, the length of packet and/or partition as well as the number of macro blocks contained in the packet can be recovered by searching for unique markers and decoding packet headers (with a large portion of information repeated), before decoding of the VLC codewords contained in the packet and partition is performed.

As an example, consider a simple VLC with an alphabet of size $K=3$, with symbols X_1, X_2, and X_3 and corresponding codewords $x_1=0$, $x_2=10$, and $x_3=11$. We assume that the probabilities of symbols are ideally matched to the code lengths l_i (i.e., that $p_i = 2^{-l_i}$). In transmitting the encoded sequence over the AWGN channel with noise standard deviation σ, we represent binary 0 by -1, and binary 1 by $+1$. Consider a 3-bit packet "110" corresponding to the symbol vector $\boldsymbol{C}=\{c(1)=3,\, c(2)=1\}$ and transmitted as $\{+1, +1, -1\}$ and received as $\boldsymbol{O}=\{0.2, 0.8, 0.05\}$. We also assume that the decoder knows the number of codewords $N=2$ *a priori* (e.g., from side information such as in a MPEG-4 error-resilient video coding bitstream).

Because the noise (and thus the observed value for each bit) is a continuous random variable, the SISO algorithm should optimize the joint probability density of packet makeup and observation, evaluated at the received observation vector, rather than the joint probability. The maximization of (2) becomes a minimization of the cumulative square error between the received and the originally transmitted values. Let $D^*(i,\, j)$ denote the global minimal cumulative square distortion of the first j bits of the packet containing exactly i codewords. Equations (1) and (2) give $\boldsymbol{C}^*(2,\, 3) = \boldsymbol{C}^*(1,\, 3-l_{c^*(2)}) \cup c^*(2)$, and

$$\begin{aligned} c^*(2) &= \arg \min_{k=1,2,3} \left\{ D^*(1,\, 3-l_k) + \left(o_{3-(l_k-1)} - x_{k,\,1}\right)^2 \right. \\ &\qquad \left. \cdots + \left(o_3 - x_{k,\,l_k}\right)^2 \right\} \\ &= \arg \min \left\{ D^*(1,\, 2) + (o_3 - x_{1,\,1})^2,\, D^*(1,\, 1) \right. \\ &\qquad + (o_2 - x_{2,\,1})^2 + (o_3 - x_{2,\,2})^2,\, D^*(1,\, 1) \\ &\qquad \left. + (o_2 - x_{3,\,1})^2 + (o_3 - x_{3,\,2})^2 \right\}. \end{aligned} \quad (4)$$

Given the observation $\mathbf{O} = \{0.2, 0.8, 0.05\}$, (4) involves finding the smaller of $D^*(1,\, 1) + \min\{[(0.8-1.0)^2 + (0.05+1.0)^2],\, [(0.8-1.0)^2 + (0.05-1.0)^2]\} = D^*(1,\, 1) + 0.9425$, and $D^*(1,\, 2) + (0.05+1.0)^2 = D^*(1,\, 2) + 1.1025$. Next, $D^*(1,\, 1)$ and $D^*(1,\, 2)$ are calculated as $D^*(1,\, 1) = (0.2+1.0)^2 = 1.44$, and

$$\begin{aligned} D^*(1,\, 2) &= \min \left\{ \left[(0.2-1.0)^2 + (0.8+1.0)^2\right], \right. \\ &\qquad \left. \left[(0.2-1.0)^2 + (0.8-1.0)^2\right] \right\} \\ &= 0.68 \end{aligned}$$

so we have $D^*(2,\, 3) = \min\{D^*(1,\, 1) + 0.9425,\, D^*(1,\, 2) + 1.1025\} = D^*(1,\, 2) + 1.1025 = 1.7825$. Therefore, the first codeword is two bits in length and is most likely x_3 by the result of the minimization for $D^*(1,\, 2)$ above. The second codeword, which contains only one bit, can only be x_1, and therefore the corresponding decoded symbol sequence is correctly identified as $\boldsymbol{C}^*(2,\, 3) = \{c^*(1) = 1,\, c^*(2) = 2\}$. In contrast with the above, when the traditional bitwise hard-decision and look-up-table based decoding is used, the hard decisions made on the received values would be $\{+1, +1, +1\}$ which would be incorrectly decoded as $\{c^*(1) = 3,\, c^*(2) = ?\}$, i.e., the decoder will report loss of synchronization at the end of the packet. For this particular example, a more intelligent "hard bit" based decoder that is able to find the "most likely" packet based on Hamming distance and *a priori* information will also fail, because for the "hard bits" $\{+1, +1, +1\}$, both the packet $\{c(1) = 1,\, c(2) = 3\}$ and the packet $\{c(1) = 3,\, c(2) = 1\}$ will have a Hamming distance of 1 to the received "hard bits." Also, because the codeword probabilities are matched to the code lengths, the decoder will not be able to find a better one from these two possibilities.

The procedure illustrated with the example is recursive. However, one can also make the optimization nonrecursive, either by using standard techniques of resolving recursive functions or by more explicitly building a trellis and optimizing on the trellis. In our simulations, we chose the recursive formulation of the algorithm, for it could be trivially implemented with a programming language such as C. Independent of the implementation, the complexity of such a decoding is linear to the size of the packet and the alphabet. It should be noted, however, to achieve this complexity in the direct recursive implementation, care needs to be taken to "remember" reduced-sized optimizations already made. As an example, consider again the optimization of (4). In (4), $D^*(1,\, 1)$ appeared twice. After the first time $D^*(1,\, 1)$ is obtained, a flag should be set and the value of $D^*(1,\, 1)$ saved, so that when the decoder needs to use $D^*(1,\, 1)$ again it does

TABLE I
COMPARISON OF SISO AND BIT-WISE HARD DECISION (HD) BASED DECODING

σ	SISO decoding		HD decoding	
(SNR)	Packet error rate	Symbol error rate	Packet error rate	Symbol error rate
0.3 (10 dB)	9.5×10^{-3}	1.9×10^{-3}	9.5×10^{-2}	3.1×10^{-2}
0.4 (8 dB)	0.49	0.12	0.71	0.24
0.5 (6 dB)	1.0	0.35	1.0	0.46

TABLE II
STATISTICS OF INVALID DECODING RESULTS FOR HD DECODING

σ (SNR)	HD decoder out-of-sync	HD outputs incorrect number of symbols		
		Probability of occurance	SISO performance for corresponding packets	
			Packet error rate	Symbol error rate
0.3 (10 dB)	5.0×10^{-4}	5.4×10^{-2}	6.6×10^{-2}	1.5×10^{-2}
0.4 (8 dB)	5.4×10^{-3}	0.58	0.61	0.15
0.5 (6 dB)	2.2×10^{-2}	0.80	1.0	0.35

not perform the recursion needed for calculating $D^*(1, 1)$ all over again. This process of remembering performed optimizations can be mapped directly to the Bellman's principal used in optimization performed on a trellis, in which case the optimal path to each intermediate node in the trellis still needs to be performed, as the need for calculating intermediate $D^*(.,.)$s, however, only the optimal path to each node is saved, and it is calculated only once.

Table I shows simulation results for packets containing $N = 100$ symbols drawn from the code $x_1 = 1, x_2 = 01, x_3 = 001, x_4 = 0001, x_5 = 00\,001, x_6 = 000\,001, x_7 = 0\,000\,001, x_8 = 00\,000\,001, x_9 = 000\,000\,001$, and $x_{10} = 100\,000\,000$ using symbol probabilities matched to the codeword length (i.e., the probability of a l-bit codeword is 2^{-l}). For each value of noise standard deviation σ, 20 packets were produced using symbols following this probability distribution, and each packet was corrupted by noise and then VLC decoded. One thousand different noise realizations were used for each packet. Packet error rate gives the fraction of packets in which one or more errors are present in the output of the VLC decoder. Symbol error rate gives the fraction of symbols which are incorrect. It is clear from the table that when the number of symbols N is known *a priori*, the SISO approach gives significantly better performance than hard decision decoding both in terms of packet and symbol error rates over a wide range of noise levels.

Another advantage of SISO decoding is that it avoids the synchronization problem inherent to hard decision VLC decoders, which do not guarantee that the final bit of a packet will coincide with the final bit of a codeword. Furthermore, in hard decision VLC decoding, correct synchronization status at the end of the packet can occur even when there are decoding errors, due to the well-known self-resynchronizing property of VLC codes. Another error that can occur in hard decision VLC decoders is an output containing an incorrect number of symbols. The synchronization and number of codewords information is what enables a hard decision based decoder to detect errors. As a matter of fact, hard decision VLC decoders are highly effective at detecting errors using such information, but they can not easily correct them. The SISO decoder, while not strictly able to perform error correction, can at least impose a constraint on the number of symbols produced during decoding, and gives the output sequence that is most probable given the observation and subject to the constraint. If one desires to use the output of a SISO decoder for error detection (e.g., to estimate the location of errors), one can compare the most likely packet with the received one.

These issues are explored in Table II, which was generated using the same simulation conditions as Table I.

The first column in Table II gives the fraction of packets for which hard decision decoding terminates in an unsynchronized state. Considering these numbers along with the hard decision symbol error rate results from Table I confirms that, even in the presence of large error rates, correct synchronization at the end of the packet can be highly probable in hard decision decoding. The second column in the table gives the fraction of packets for which the hard decision decoder terminates in a synchronized state but outputs an incorrect number of symbols. The third and fourth columns in the table explore the performance of the SISO decoder on these packets. This sheds light on the performance of the SISO decoder on packets which would lead to (typically uncorrectable) error indications by a hard decision decoder. The improvement shown by SISO decoding is significant. For example, for an SNR of 8 dB, the table shows that for hard decision decoding, 58% of the packets in error would be due to an incorrect number of symbols. In SISO decoding, the packet error rate is only 61%, meaning that 39% of the packets would

be decoded with no error at all (correct number of symbols and all symbols correct). The symbol error rate is only 15%.

In addition to using soft information and picking the "optimal" codeword concatenation using *a prior* knowledge of the symbol distribution and synchronization information, the algorithm presented also outputs the "soft" likelihood information associated with the optimal output. This information could used in cases such as those when L and/or N are unknown. In such cases, assumptions on the values of N and/or L can be made, and the optimal decoding result for each N and/or L can be obtained using the algorithm presented here. Then, the "soft" likelihood of the optimal output associated with each N and/or L assumption can be use to pick one best decoding output. More discussions on this approach with simulation results can be found in [1].

III. Conclusion

We have presented an algorithm for using soft information in VLC decoding. Simulations demonstrate a significant improvement over traditional VLC decoding based on hard inputs. The method here exploits not only the presence of soft values, but also *a priori* knowledge of the number of bits L and the number of symbols N. This also helps to improve the performance with respect to hard decision VLC methods, which offer no means to enforce consistency between the expected and actual number of symbols produced by the decoder. The techniques presented here can be combined with source probability estimation methods to reduce or eliminate the performance penalty due to mismatch.

References

[1] J. Wen and J. Villasenor, "Utilizing soft information in decoding variable length codes," in *Proc. IEEE Data Compression Conf.*, Snowbird, UT, Mar. 1999, pp. 131–139.

[2] V. Buttigieg, "Variable-length error-correcting codes," Ph.D. dissertation, Univ. of Manchester, Manchester, U.K., 1995.

[3] M. Park and D. Miller, "Improved joint source-channel decoding for variable-length encoded data using soft decisions and MMSE estimation," in *Proc. IEEE Data Compression Conf.*, Snowbird, UT, Mar. 1999, p. 544.

[4] A. H. Murad and T. E. Fuja, "Robust transmission of variable-length encoded sources," in *Proc. IEEE Wireless Comm. and Networking Conf.*, vol. 2, Sept. 1999, pp. 968–971.

[5] R. Bauer and J. Hagenauer, "Iterative source/channel-decoding using reversible variable length codes," in *Proc. IEEE Data Compression Conf.*, Snowbird, UT, Mar. 2000, pp. 93–102.

[6] ——, "On variable length codes for iterative source/channel decoding," in *Proc. IEEE Data Compression Conf.*, Snowbird, UT, Mar. 2001.

[7] M. P. C. Fossorier, F. Burkert, S. Lin, and J. Hagenauer, "On the equivalence between SOVA and max-log-MAP decodings," *IEEE Commun. Lett.*, vol. 2, pp. 137–139, May 1998.

[8] S. Benedetto, D. Divsalar, G. Montorsi, and F. Pollara, "A soft-input soft-output APP module for iterative decoding of concatenated codes," *IEEE Commun. Lett.*, vol. 1, pp. 22–24, Jan. 1997.

[9] R. E. Bellman, *Dynamic Programming and Modern Control*. New York: Academic, 1966.