An Integrated Joint Source-Channel Decoder for MPEG-4 Coded Video[1]

Qingyu Chen
Department of Electrical and
Computer Engineering
Stevens Institute of Technology
Hoboken, NJ, 07030
Email: qchen1@stevens-tech.edu
Phone: 201-216-5615 Fax: 201-216-8246

K.P. Subbalakshmi
Department of Electrical and
Computer Engineering
Stevens Institute of Technology
Hoboken, NJ, 07030
Email: ksubbala@stevens-tech.edu
Phone: 201-216-8641 Fax: 201-216-8246

***Abstract*— The MPEG-4 video coding standard uses variable length codes (VLCs) to encode the indices of intra and inter macroblocks after discrete cosine transform and quantization. Although VLCs can achieve good compression, they are very sensitive to channel errors. Joint source-channel decoding (JSCD) is emerging as an efficient method for dealing with this sensitivity to channel errors. This paper proposes an integrated joint source-channel decoder (I-JSCD) for first order Markov sources coded with Huffman codes and convolutional codes over a binary symmetric channel. The proposed decoder combines the source state space of the Huffman decoder and channel state space of the Viterbi decoder together to construct a joint decoder with a Viterbi-like structure. We applied this I-JSCD to VLCs of both inter and intra macroblocks in MPEG-4 coded video. Experiments indicate that the proposed decoder gives significant improvements (maximum 7 dB) than a separate scheme, where a constrained joint source-channel decoder is concatenated with a Viterbi decoder at the same rate.**

I. INTRODUCTION

With the benefit of increasing bandwidth in the Internet and wireless networks, many new video communications applications like Internet video streaming and mobile visual phone, are becoming more and more popular. Currently, most of the video coding standards, like MPEG-4, use *variable length codes* (VLC). Although VLCs provide good compression efficiency, they are very sensitive to channel errors. A single bit error can cause the decoder to parse the codeword boundaries incorrectly leading to a loss in synchronization. Such loss of synchronization adversely affects the reconstructed video quality.

The MPEG-4 standard provides a set of tools to deal with the error resilience problem which include: *reversible variable length codes* (RVLC) [1], [2], *resynchronization markers* and *data partitioning* [3]. Although these error resilience tools can protect the video stream from some channel errors and can recover some corrupted video frames; they can only restrict the propagation of errors to a small region of time or space and not eliminate the errors completely. Therefore, when the channel bit error rate is very high, the decoded video sequence will still have many corrupted blocks, which could potentially degrade the visual quality significantly.

Joint source-channel coding (JSCC) is regarded as a good alternative for achieving reliable communication of signals across noisy channels. The rationale behind using JSCC is that in practical systems, Shannon's source-channel separation theorem [4] does not hold under delay and complexity constraints. JSCC tries to design the source coder and channel coder in some joint fashion, so as to provide better error protection and bandwidth utilization. JSCC schemes can be broadly classified into three different categories: *joint source-channel encoding* (JSCE) [5], [6], *joint source-channel decoding* (JSCD) [7]-[14], and *rate allocation* strategies [15]. As the names suggest, these deal with the joint design of encoders, decoders and the rate allocation between the channel and source codes respectively. JSCD uses the redundancy remaining in the compressed stream after source coding to reduce the effect of the channel noise in the decoded signal. JSCD schemes can be further classified into *constrained* JSCD (C-JSCD) [7]-[11] and *integrated* JSCD (I-JSCD) [12]-[14]. C-JSCDs are typically source decoders that are built using prior knowledge of channel characteristics while I-JSCDs combine the source and channel decoder into one unit. Examples of C-JSCDs for sources with memory include [7]-[11]. In [7], a source-symbol synchronized Viterbi decoder was designed for convolutional and VLC coded sources. Authors in [8] developed exact and approximate MAP decoder for variable length encoded data transmitted over BSC. We recently developed a novel state-space structure and designed a *maximum a posteriori probability* (MAP) decoder for VLC encoded memoryless sources [9] and Markov sources [10], [11] transmitted over *binary symmetric channels* (BSCs) for the case where the number of transmitted words is not known to the decoder.

Some researchers have proposed iterative JSCDs, where the VLC decoder and the soft channel decoder are serially concatenated and the decoding is done iteratively; e.g. the work on symbol by symbol MAP decoding for VLC coded memoryless source [16]. Hedayat *et al* compared an iterative JSCC with a separable system under the same overall rate and computational complexity [17]. Kliewer *et al* designed

[1]This work was supported in part by The Stevens Center for Wireless Network Security and The New Jersey Center for Telecommunications.

an iterative decoder consisting of an *a posteriori* probability based channel decoder and an *a posteriori* probability based VLC source decoder [18].

In [12], an I-JSCD was developed for convolutional encoded Huffman codes by appropriately combining the graph representation of the Markov source, the Huffman source decoder, and the convolutional channel decoder. This algorithm was then used to exploit the residual redundancy in *motion vectors* (MVs) [13]. Lakovic and Villasenor proposed a modified Viterbi decoding trellis [14], incorporating information about the structure of the Huffman source code. This decoder differs from [12] in that it is designed for memoryless sources. The authors then incorporated it into a turbo decoder in [19], giving better performance.

In this paper we design a MAP based I-JSCD for VLC coded first order Markov sources, convolutional encoded for transmission over a BSC. The proposed decoder combines the source and channel state space to construct a delayed decision, joint decoder. Our joint state space is smaller than that of [12] and thus has less computational complexity. Our work differs from [14] in that we consider a first order Markov source rather than a memoryless source.

As we showed in [20], [21], the VLC coded inter and intra macroblocks (MBs) in MPEG-4 coded video can be modelled as 1-D Markov processes. Hence, in this paper we apply the proposed I-JSCD for the MPEG-4 source and then compare the results with a system that uses the C-JSCD developed in [11], concatenated with a Viterbi decoder.

II. Problem Formulation

We define our I-JSCD problem as a MAP decision problem, which is equivalent to minimizing the probability of decision error for the sequence. In this paper we generalize the constrained MAP decoder developed in [11] to include the decoder for the convolutional code as well. In other words, we design a MAP based *integrated* joint source-channel decoder. Mathematically, this problem can be expressed by Equation 1, where $t_{\hat{j}}$ is the most probable transmitted stream after convolutional coding from all possible transmitted streams $\{t_j\}$ and r_i is the received stream.

$$t_{\hat{j}} = \arg\max_{t_j} \Pr(r_i|t_j)\Pr(t_j) \tag{1}$$

Let j^{th} sequence from the set of all possible VLCs sequences of a first order Markov source be denoted by c_j; and let t_j represent the stream obtained from convolutional encoding c_j. Note that there is an one-to-one mapping between c_j and t_j. Hence, the most probable VLCs stream $c_{\hat{j}}$ is calculated as:

$$c_{\hat{j}} = \arg\max_{c_j} \Pr(r_i|t_j)\Pr(c_j) \tag{2}$$

Since the source is 1-D Markov and the channel is BSC, we can break the source term $\Pr(c_j)$ and channel term $\Pr(r_i|t_j)$ into smaller terms. Let $c_j = \{c_{j,k}\}_{k=1}^{n(j)}$, where $c_{j,k}$ is the codeword corresponding to the k^{th} symbol in the VLC stream and $n(j)$ is the total number of codewords in this sequence. Let $t_{j,k}$ denote the convolutional coded symbol corresponding to $c_{j,k}$ and l_k is the length of $t_{j,k}$ in bits. Similarly, let $r_{i,k}$ be the k^{th} symbol of the received bitstream under the same partition as t_j. Then Equation 2 becomes:

$$c_{\hat{j}} = \arg\max_{c_j} \Big[\Pr(c_{j,1})\epsilon^{d_H[t_{j,1},r_{i,1}]}(1-\epsilon)^{(l_1-d_H[t_{j,1},r_{i,1}])} \times \prod_{k=2}^{n(j)} \Pr(c_{j,k}|c_{j,(k-1)})\epsilon^{d_H[t_{j,k},r_{i,k}]}(1-\epsilon)^{(l_k-d_H[t_{j,k},r_{i,k}])}\Big] \tag{3}$$

Here $\Pr(c_{j,1})$ is the probability that codeword $c_{j,1}$ was transmitted first, $\Pr(c_{j,k}|c_{j,k-1})$ is the probability that the codeword $c_{j,k}$ was transmitted immediately after the codeword $c_{j,k-1}$, $d_H[s_{j,k}, r_{i,k}]$ is the Hamming distance between the transmitted symbols $s_{j,k}$ and the corresponding received symbols $r_{i,k}$. This maximization over j, effectively searches through all possible error sequences and bit stream partitions. This process can be cast as a dynamic programming problem once an appropriate state-space and the corresponding trellis are defined. We develop the state-space in Section III.

III. Integrated Source-Channel State Space

In [11], [22], we developed a state-space for the C-JSCD on the Huffman coded Markov source. This state-space consisted of complete and incomplete states corresponding to whether the decoder had just received the last bit of a codeword or was in the middle of a codeword. The complete state consisted of all the Huffman codewords. For our current problem, we start with two state-spaces: one for the Huffman coded Markov source and one for the channel code. These are then combined appropriately to give the overall state-space for the I-JSCD. The source state-space now contains all internal nodes (other than the root) of the Huffman tree as well. These correspond to the incomplete states, whereas the leaves correspond to the complete states.

Figure 1(a) illustrates the source state-space for the Huffman code $\mathcal{C} = \{0, 10, 110, 111\}$. The complete states are $a(0), b(10), c(110), d(111)$ and the states $e(1)$ and $f(11)$ are the incomplete states. These source states may further be divided into two groups according to whether the rightmost bit in these states is a '1' (d, e and f) or a '0' (a, b and c). This classification essentially corresponds to the input bit from the channel decoder to the source decoder.

The channel state-space is simply the state-space associated with the convolutional code. Figure 1(b) shows a channel decoder state-space for the $1/3$ convolutional code given in [23]. The states here correspond to the contents of the shift register: $s_0 \rightarrow 00, s_1 \rightarrow 10, s_2 \rightarrow 01, s_3 \rightarrow 11$. These states can also be divided into two groups corresponding to whether the channel decoder output is a '1' (s_1, s_3) or a '0' (s_2, s_4).

Since the output of the channel decoder is the input of the source decoder, we can combine the corresponding groups in the source state space and the channel state space. The integrated state space for the example Huffman and convolutional codes is given in Figure 1(c). Each state in this state space reflects the current status of the joint decoder. For example,

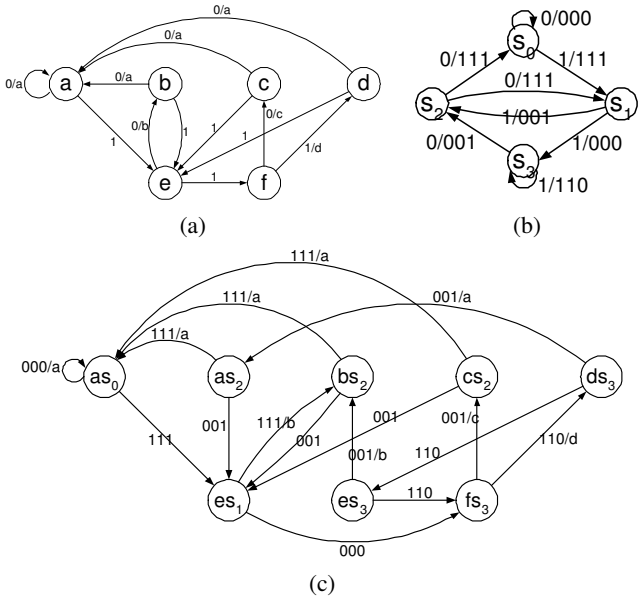

Fig. 1. Combing source and channel state space. (a) source state space. (b) channel state space. (c) integrated state space.

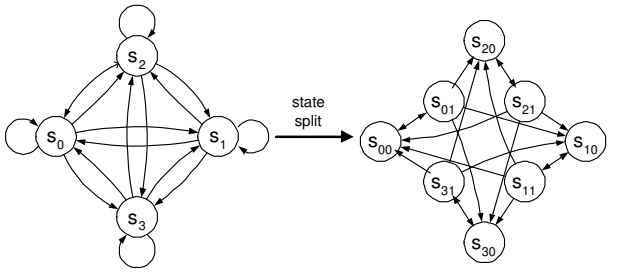

Fig. 2. Splitting the channel states.

state as_0 means the decoder is in state s_0 and currently producing codeword a. Note that the size of the integrated state space is reduced in this diagram. States (bs_0, cs_0, ds_1, fs_1) are eliminated because they can never be reached by other states. Take bs_0 as an example, from Figure 1(a) and (b), we see that bs_0 can only be reached from a state combing source state e and channel state s_0 or s_2. However, such combinations are not possible because they are in the different groups.

While the above procedure works for a k/n convolutional code where $k = 1$, channel splitting needs to be done for $k > 1$. In this case, there are multiple output symbols for each transition between channel states. However, there is only one input symbol for each source state transition. So, to combine the source and channel state, each channel state has to be split into k sub-states. An example is given in Figure 2. The state space of a rate $2/3$ convolutional code has four channel states $(s_0 \sim s_3)$. Each state s_i is split into two sub-states s_i^0 and s_i^1. There are only two types of transitions: from s_i^0 to s_i^1 or from s_j^1 to s_i^0. For each of these, there is only one symbol produced, which enables the channel state space to be combined with the source state space. In this example, as usual we can divide

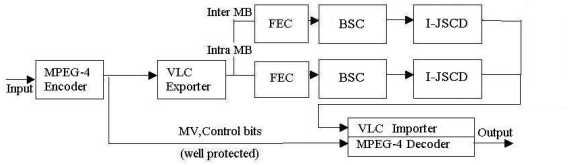

Fig. 3. Experimental Set-up for MPEG-4 video

these sub-states into two groups and combing them with the source states in the same group. Using channel state split, we can combine the state space of any k/n convolutional code with that of any Huffman code.

From the point view of the source state representation, Murad's algorithm [12] needs $N_{\text{leaf}} \times (N_{\text{internal}} + 1)$ states where N_{leaf} is the number of leaves in the Huffman tree and N_{internal} is the number of internal nodes in the Huffman tree (except root). Our algorithm needs $N_{\text{leaf}} + N_{\text{internal}}$ states. It can be shown that when $N_{\text{internal}} \geq 1$, $N_{\text{leaf}} + N_{\text{internal}} < N_{\text{leaf}} \times (N_{\text{internal}} + 1)$. For the example in Figure 1(a), Murad's algorithm needs 12 source states while ours only need 6 states.

IV. Experimental Setup and Results

In our experiments, we encode the MPEG-4 inter and intra coded MBs using VLCs and decode them with the proposed I-JSCD developed for VLCs over BSCs. For comparison, results of a C-JSCD [11] and a traditional MPEG-4 decoder preceded by the usual Viterbi decoder (for the convolutional code) are also presented. The experimental set-up is depicted in Figure 3. The bit streams corresponding to inter and intra MB are corrupted separately through eight instances of the BSC (simulated using different random seeds). The Microsoft MPEG-4 VM encoder and decoder program are used for all our experiments and the data partitioning and resynchronization markers are deployed in all experiments. The remaining portions of the MPEG-4 video stream (MV, control bits, etc) are assumed to be very well protected using strong forward error correcting schemes. Twelve frames from the Susie sequence containing one object are encoded as 2 I-frames (the 1^{st} and 7^{th},) and 10 P-frames. We choose frames 42^{nd} to 53^{rd} from the total (150 frames) because this section exhibits comparatively higher activity. The source rate of the encoded sequences for all experiments are kept the same at 0.3 bits per pixel. The same video sequence is used in the training step to get the probability distribution. Although the video bit-stream can be highly non-stationary, we can potentially estimate the probability of the stream on the fly for smaller segments and update the decoder with the new statistics. Different channel bit error rates (ϵ) in the $[10^{-4}, 10^{-1.5}]$ range are simulated to corrupt the encoded stream. The results presented here are the average of eight realizations of the channel, corresponding to eight different error patterns. The rate $2/3$ convolutional encoder in [23] was used as the channel code.

Figures 4 provides a frame-by-frame comparison of the

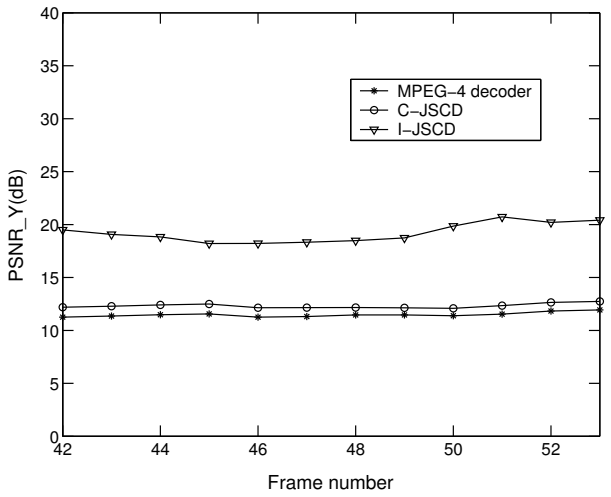

Fig. 4. PSNR of Y component for the Susie sequence at $\epsilon = 10^{-1.5}$.

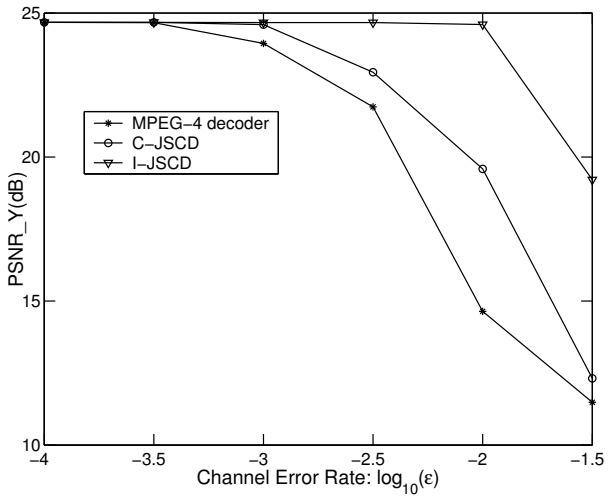

Fig. 5. Average PSNR of Y component of the Foreman sequence.

performance among proposed I-JSCD, the C-JSCD in [11] and the MPEG-4 decoder in terms of the average PSNR for the Y component of the decoded video frames. The channel error rate is fixed at $10^{-1.5}$ for this plot. We see that the performance of the proposed decoder is significantly superior to the standard MPEG-4 decoder and the C-JSCD. The average improvement over the C-JSCD is 6.5 dB.

Figure 5 plots the average PSNR of the Y component of the whole video sequence at different channel bit error rates. From this figure, we see that the I-JSCD performs much better than the C-JSCD and the standard MPEG-4 decoder when ϵ is between $10^{-3.0}$ and $10^{-1.5}$. When $\epsilon < 10^{-3.0}$, the performance of the I-JSCD is almost the same as the C-JSCD. This is because at low error rates, the convolutional codes and C-JSCD can correct almost all the channel noise and hence there is little to be gained by using the I-JSCD. Finally, in Figure 6 we observe the actual I-JSCD, C-JSCD

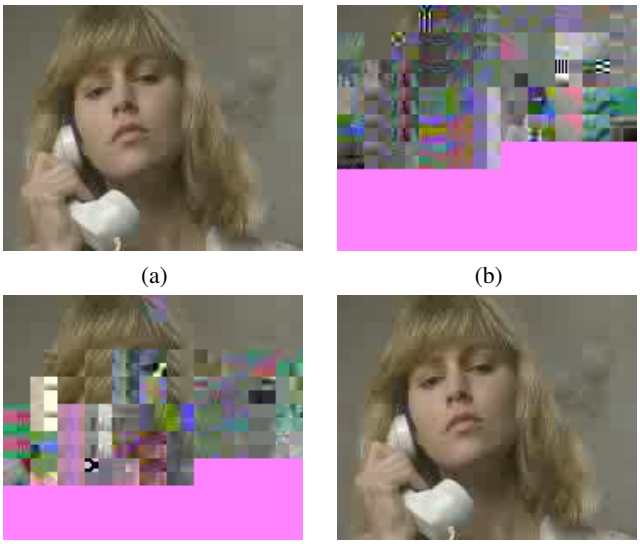

Fig. 6. First frame of the Susie sequence. (a) MPEG-4 decoder decoded frame without channel error. (b) MPEG-4 decoder decoded frame at $\epsilon = 10^{-2.0}$. (c) C-JSCD decoded frame at $\epsilon = 10^{-2.0}$. (d) I-JSCD decoded frame at $\epsilon = 10^{-2.0}$.

and MPEG-4 decoded frames for the Susie sequence when $\epsilon = 10^{-2.0}$, giving a more perceptual idea of the performance of the decoders. The decoded frame without channel errors is also provided. The I-JSCD seems to correct most of the block errors caused by the channel, which the C-JSCD decoder can not salvage.

V. Conclusions

This paper presented an optimal integrated joint source-channel MAP decoder for variable length encoded 1-D Markov source. The proposed decoder was then applied to VLC coded inter MB and intra MB of a MPEG-4 video stream over a BSC. Simulation results demonstrate that this I-JSCD does significantly better than both the system comprising of the constrained MAP decoder followed by the Viterbi decoder and the conventional MPEG-4 decoder at various error rates.

References

[1] J. Wen and J. D. Villasenor, "Reversible variable length codes for efficient and robust image and video coding," in *IEEE Data Compression Conference*, 1998, pp. 471–480. [Online]. Available: citeseer.nj.nec.com/wen98reversible.html

[2] Y. Takishima, M. Wada, and H. Murakami, "Reversible variable length codes," *IEEE Transactions on Communications*, vol. 43, pp. 158–162, 1995.

[3] Y. Wang, S. Wenger, J. Wen, and A. K. Katsaggelos, "Error resilient video coding techniques," *IEEE Signal Processing Magazine*, vol. 17, pp. 61–82, July 2000.

[4] C. E. Shannon, "The mathematical theory of communication," *Bell Sys. Tech. Journal*, vol. 28, pp. 379–423, Oct. 1949.

[5] N. Farvardin, "A study of vector quantization for noisy channels," *IEEE Transactions on Information Theory*, vol. 36, pp. 799–809, July 1990.

[6] V. Buttigieg and P. G. Farrell, "Variable-length error-correcting codes," *IEE Proceedings in Communications*, vol. 147, no. 4, pp. 211–215, Aug. 2000.

[7] N. Demir and K. Sayood, "Joint source/channel coding for variable length codes," in *IEEE Data Compression Conference*, Snowbird, UT, 1998, pp. 139–148.

[8] M. Park and D. J. Miller, "Joint source-channel decoding for variable-length encoded data by exact and approximate MAP sequence estimation," *IEEE Transactions on Communications*, vol. 48, pp. 1–6, 2000.

[9] K. P. Subbalakshmi and J. Vaisey, "Optimal decoding of entropy coded memoryless sources over binary symmetric channels," in *IEEE Data Compression Conference*, Mar.-Apr. 1998, p. 573.

[10] ——, "Joint source-channel decoding of entropy coded Markov sources over binary symmetric channels," in *IEEE International Conference on Communications*, vol. 1, June 1999, pp. 446–450.

[11] ——, "On the joint source-channel decoding of variable-length encoded sources: the BSC case," *IEEE Transactions on Communications*, vol. 49, pp. 2052–2055, Dec. 2001.

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

[13] ——, "Exploiting the residual redundancy in motion estimation vectors to improve the quality of compressed video transmitted over noisy channels," in *Proc. Int. Conf. Image Processing*, Oct. 1998.

[14] K. Lakovic, J. Villasenor, and R. Wesel, "Robust joint Huffman and convolutional decoding," in *IEEE Vehicular Technology Conference*, 1999, pp. 2551–2555.

[15] M. Bystrom and J. W. Modestino, "Combined source-channel coding for transmission of H.263 coded video with trellis-coded modulation over a slow fading Rician channel," in *IEEE International Symposium on Information Theory*, Aug. 1998, p. 12.

[16] R. Bauer and J. Hagenauer, "Symbol by symbol MAP decoding of variable length codes," in *3rd ITG Conference Source and Channel Coding*, Munich, Germany, Jan. 2000.

[17] A. Hedayat and A. Nosratinia, "On joint iterative decoding of variable-length codes and channel codes," in *IEEE Conference on Communications, Control, and Computing*, Oct. 2002.

[18] J. Kliewer and R. Thobaben, "Combining FEC and optimal soft-input source decoding for the reliable transmission of correlated variable-length encoded signals," in *IEEE Data Compression Conference*, 2002, pp. 83–91.

[19] K. Lakovic and J. Villasenor, "Combining variable length codes and turbo codes," in *IEEE Vehicular Technology Conference*, 2002, pp. 1719–1723.

[20] Q. Chen and K. P. Subbalakshmi, "Trellis decoding for MPEG-4 streams over wireless channels," in *Proc. SPIE Electronic Imaging: Image and Video Communications and Processing*, Jan. 2003, pp. 810–819.

[21] ——, "Joint source-channel decoding for MPEG-4 video transmission over wireless channels," in *IEEE Journal on Selected Areas in Communications-Special Issues on Recent Advances in Wireless Multimedia. to appear*.

[22] K. P. Subbalakshmi and J. Vaisey, "On the joint source-channel decoding of variable-length encoded sources: The additive Markov channel case," *IEEE Transactions on Communications, (To Appear)*.

[23] J. G. Proakis, *Digital Communications: Fourth Edition*. New York, NY: McGraw-Hill, 2001.