# DISTRIBUTED SOURCE CODING WITH CONTEXT MODELING

*Yong Sun*
Electrical and Computer Engineering, Texas A&M Univ.,
11B/C Zachry Bld., College Station, TX 77843
Email: sunny@ee.tamu.edu

*Jin Li*
Communication and Collaboration Systems, Microsoft Research
One Microsoft Way, Bld. 113, Redmond, WA 98052
Email: jinl@microsoft.com

## ABSTRACT

We introduce context modeling into the distributed source coding (DSC). By forming contexts from prior coded bitplanes of both the reference and DSC coded peer, we split the virtual channel between the two correlated bitplanes into several virtual sub-channels of different characteristics. The selection of the sub-channel becomes side information that is known to the receiver. As a result, the DSC coding bitrate is reduced. We also investigate a number of practical implementation issues in DSC; e.g., the use of turbo-based channel code vs. the LDPC-based channel code, and the use of a random flipper to handle the binary asymmetric channel. We implemented a practical DSC audio coding system. We show that DSC without context modeling shows little performance gain compared with separate source coding with context modeling. In comparison, context DSC achieves an overall rate saving of 36%, even as we only applied DSC on a selected number of bitplanes, and considering the channel code loss in practical implementation.

## 1. INTRODUCTION

Distributed source coding (DSC) [1][2] enables efficient compression of the outputs of two or more physically separated sources without the sources communicating with each other. DSC has become an active research area because it enables efficient information compression for a number of emerging applications, e.g., CEO (central estimation officer) problem in a sensor network, the peer-to-peer (P2P) streaming of analog TV/radio, etc.

The information theory foundation of DSC is the Slepian-Wolf Theory [3], which states that the lossless compression of two separated source can be made as efficient as if they are compressed together as long as joint decoding is done at the receiver. As a result, DSC is often referred as Slepian-Wolf coding (SWC). Although the theory behind DSC has been well understood for over 30 years, practical DSC implementation is far from simple. The implementation of DSC is closely tied to channel coding [1]. The reference and the coded symbol of DSC can be considered as the input and the output of a virtual channel, and the DSC design is equivalent to the design of a proper channel code that is capable of correcting errors of the virtual channel. As a result, direct DSC coding on multi-level symbols can be converted to the problem of designing an efficient channel code that may handle multi-level channel errors. Such a channel code has not been well studied in the past, and practical implementation of such code often results in performance far inferior to the channel code designed for the well-understood binary symmetric channel (BSC), e.g., turbo codes [4] and LDPC codes [5]. A popular DSC implementation for multi-level symbols is based on the bitplane coding. In such scheme, coefficient at the reference and the coded peers are first separated into bitplanes. Then, DSC is applied to each bitplanes separately. The approach converts a multi-level DSC into multiple binary DSCs.

Although simple and straightforward, the practical implementation of bitplane based DSC does not demonstrate superior compression performance compared to separately entropy encoding the two sources. This is due to a number of factors. First, bitplane source coding usually uses advanced statistical technology such as context modeling, which takes advantage of the correlation between the bits. No existing DSC has used context modeling or has effectively explored the correlation between the bits. Second, existing channel code designed for BSC has a number of limitations, e.g., it usually targets BSC channel, has only a number of code profiles and may have to use a higher code rate for a certain effective channel error rate in DSC. Moreover, the performance gap between what can be achieved by a practical channel coder and the theoretical bound is also larger than the performance gap between what can be achieved by a practical source coder and its theoretical bound.

In this work, continuing on the footsteps of the practical DSC implementation pioneered by Aaron and Girod in [6] and Xiong *et al.* in [7], we work further to bring DSC closer to practicality. Two issues have been investigated in the paper. First, we introduce context modeling in DSC coding, which allows correlation between the bitplane to be explored. Second, we investigate a number of practical channel code issues that effectively adapt the channel code designed for BSC to DSC coding scenario. The rest of the paper is organized as follows. We introduce our DSC system in Section 2. We discuss the context DSC and practical channel code for DSC in Section 3. The efficiency of our proposed system is evaluated by DSC audio coding in section 4. We give a conclusion in Section 5.

## 2. DISTRIBUTED SOURCE CODING: FRAMEWORK

The DSC system framework is shown in Figure 1. We focus on the coding with side information. The upper path of the system is a traditional source coder. The input signal $\mathbf{Y_1}$ is first transform and quantized to coefficients $\mathbf{P_1}$. The quantized coefficients are then split into bitplanes, and entropy encoded from the most significant bitplanes (MSB) to the least significant bitplanes (LSB). At the receiver, the compressed bitstreams are entropy decoded. Because the entropy encoder and decoder module is lossless, if all bitplanes are decoded, the recovered quantized coefficients will be exactly the same as $\mathbf{P_1}$. Finally, the decoded coefficients are inverse quantized and inverse transformed to reconstruct the source $\hat{\mathbf{Y}}_1$.

The lower path of the system is the DSC path. The input signal of the lower path $\mathbf{Y_2}$ is transformed and quantized into coefficients $\mathbf{P_2}$ via exactly the same transform and quantization module of the upper path. The coefficients are split into bitplanes, and DSC

encoded from MSB to LSB. Obviously, the correlation between the DSC bitplane and its reference bitplane is stronger at the more significant bitplanes, and is weaker at the less significant bitplanes. At the decoder, the bitplane is DSC decoded with the corresponding bitplane of the upper path as side information. Finally, the DSC decoded bitplanes are reassembled into the quantized coefficients $\mathbf{P_2}$, inversely quantized and inversely transformed to recover the transform coefficients $\hat{\mathbf{Y}}_2$.

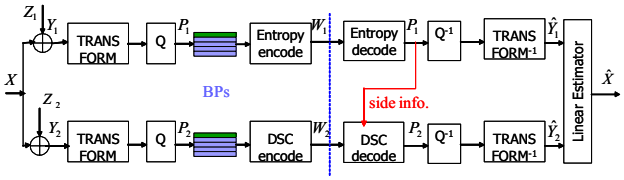


Figure 1 System framework of direct and indirect distributed source coding.

In Figure 1, both the entropy coding and the DSC coding proceeds from MSB to LSB. Moreover, coding of the less significant bitplanes only requires information of the more significant bitplanes. As a result, our proposed DSC system is scalable. The bitstreams of both paths can be truncated for decoding of signals at less quality level.

The above framework assumes that both signals $\mathbf{Y_1}$ and $\mathbf{Y_2}$ are observed from separate sensors. It is called direct DSC coding. If both signals come from a common signal source $\mathbf{X}$, $\mathbf{Y_1}$ and $\mathbf{Y_2}$ can be considered the noisy observation of the source. The problem is called indirect DSC coding, and the source $\mathbf{X}$ is not observed directly. We may estimate the source via a linear estimator:

$$\hat{\mathbf{X}} = \alpha_1 \hat{\mathbf{Y}}_1 + \alpha_2 \hat{\mathbf{Y}}_2, \text{ where} \quad (1)$$

$$\alpha_1 = \alpha_2 = \frac{E\left[\mathbf{X}^2\right]}{2E\left[\mathbf{X}^2\right] + \left(\sigma_z^2 + D\right)}. \quad (2)$$

In eqn (2), $E[\mathbf{X}^2]$ is the energy of the source, $\sigma_z^2$ is the observation noise variance, and D is the coding distortion of the signal $Y_1$ and $\mathbf{Y_2}$. The observation error of the source $\mathbf{X}$ can be calculated as:

$$E\left[\left(\mathbf{X} - \hat{\mathbf{X}}\right)^2\right] = \frac{E\left[\mathbf{X}^2\right]\left(\sigma_z^2 + D\right)}{2E\left[\mathbf{X}^2\right] + \left(\sigma_z^2 + D\right)}. \quad (3)$$

## 3. PRACTICAL DISTRIBUTED SOURCE CODING

### 3.1. Context modeling in distributed source coding

Context modeling is a widely used technology in modern source coding. It has been extensively studied for statistical model-based arithmetic coding, e.g., [8]. JPEG 2000, a modern image compressor, utilizes context modeling in its bitplane coding engine to improve the compression efficiency. Context is defined as a category value assigned to a certain symbol (bit) based on prior coded symbols (bits). In JPEG 2000, the bits in a bitplane are classified into three big categories based on the context: significant identification, refinement and sign. Sign is just + and - of the coefficient. For a certain bit of a certain coefficients, if all the bits in the more significant bitplanes are all 0s, the bit belongs to the significant identification. Otherwise, it belongs to refinement. An example is shown in Figure 2. We put the sign bits in the topmost row. All bits in significant identification are marked with shadow, the refinement bits are shown in clear box. Context modeling can improve compression efficiency because bits belong to different context are highly different in entropy statistics and coding distortion statistics. By separating bits into different contexts (JPEG 2000 actually creates further sub-contexts, and uses 9 contexts for significant identification bit coding, 5 contexts for sign coding, and 3 contexts for refinement coding), a modern source coder may separate bits into a compound source with different entropy characteristics, apply statistical modeling separately, and apply suitable entropy coding parameter for each source.

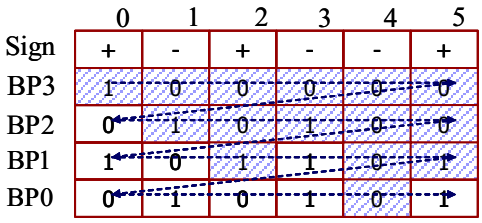


Figure 2 Context in Bitplane source coding: Significant identification bits, refinement bits, and sign bits.

In this paper, we extend context modeling to DSC. Because there are two peers in DSC, the DSC context also involve the bitplanes of both peers. Let the current coded bitplane be the *i*th bitplane, let the reference bitplane at peer 1 be $\mathbf{U_i}$, and the DSC coded bitplane at peer 2 be $\mathbf{V_i}$. We use j index to index the coefficient, therefore, the *i*th bit at peer 1 and 2 is $U_{i,j}$ and $V_{i,j}$, respectively. All more significant bitplanes at peer 1 form the reference context:

$$\mathbf{A_i} = \{\mathbf{U_{M-1}}, \cdots, \mathbf{U_{i+1}}\}, \quad (4)$$

and all the more significant bitplanes at peer 2 form the coded context:

$$\mathbf{B_i} = \{\mathbf{V_{M-1}}, \cdots, \mathbf{V_{i+1}}\}. \quad (5)$$

With context modeling, the virtual channel in DSC becomes a compound virtual channel, as shown in Figure 3. Depending on the context value of $\mathbf{A_i}$ and $\mathbf{B_i}$, a sub-channel is selected among the compound virtual channel. Different sub-channel may exhibit different channel error characteristics.

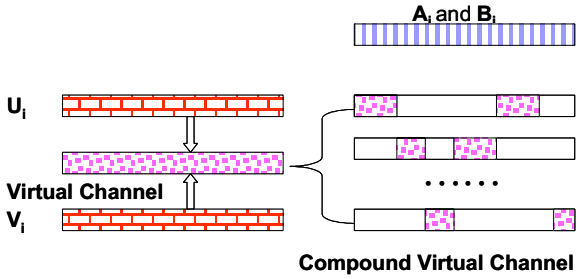


Figure 3 Context in distributed source coding.

We do observe that context modeling in DSC differs from that of source coding. First, obtaining statistics of each sub-channel is trickier. In context source coding, statistics of the already coded symbol in a context is directly used as the statistics of the symbol for future coding. This could be used similarly in DSC. However, because the correlation between the reference and coded bitplanes in DSC may change dramatically across different bitplanes, it may not as effective. An alternative approach assumes knowledge of the correlation of the reference and coded signal $\mathbf{Y_1}$ and $\mathbf{Y_2}$. We may then use Monte-Carlo simulation to calculate the statistics of each context of each bitplane. In this work, we use four contexts for bitplane coding and two contexts for sign coding (as signs are only coded when a coefficient becomes non-zero). The definition and

the statistics collected for each context are shown in Table 1 and Table 2.

Table 1 Context for bitplane in distributed source coding

| Bit context | definition | Statistics collected |
|---|---|---|
| sig,sig | $A_{i,j}=0, B_{i,j}=0$ | $p(V_{i,j}\mid U_{i,j},A_{i,j}=0,B_{i,j}=0),p(A_{i,j}=0,B_{i,j}=0)$ |
| sig, ref | $A_{i,j}=0, B_{i,j}\neq 0$ | $p(V_{i,j}\mid U_{i,j},A_{i,j}=0,B_{i,j}\neq 0),p(A_{i,j}=0,B_{i,j}\neq 0)$ |
| ref,sig | $A_{i,j}\neq 0, B_{i,j}=0$ | $p(V_{i,j}\mid U_{i,j},A_{i,j}\neq 0,B_{i,j}=0),p(A_{i,j}\neq 0,B_{i,j}=0)$ |
| ref,ref | $A_{i,j}\neq 0, B_{i,j}\neq 0$ | $p(V_{i,j}\mid U_{i,j},A_{i,j}\neq 0,B_{i,j}\neq 0),p(A_{i,j}\neq 0,B_{i,j}\neq 0)$ |

Table 2 Context for sign coding in distributed source coding

| Sign ctx | definition | Statistics collected |
|---|---|---|
| Sig | $A_{i,j}=0, B_{i,j}\neq 0$ | $p(V_{i,j}\mid U_{i,j},A_{i,j}=0,B_{i,j}\neq 0),p(A_{i,j}=0,B_{i,j}\neq 0)$ |
| Ref | $A_{i,j}\neq 0, B_{i,j}\neq 0$ | $p(V_{i,j}\mid U_{i,j},A_{i,j}\neq 0,B_{i,j}\neq 0),p(A_{i,j}\neq 0,B_{i,j}\neq 0)$ |

In DSC, the exact contexts are only available to the receiver, and are not available to the sender. As a result, the context DSC encoder only uses collective information of the compound channel. More specifically, for the DSC encoder, we will only need the channel error rate $R_i$ at bitplane $i$:

$$\begin{aligned} R_i &= H\left(V_{i,j}\mid U_{i,j},A_{i,j},B_{i,j}\right) \\ &= H\left(V_{i,j}\mid U_{i,j},A_{i,j}=0,B_{i,j}=0\right)\cdot p\left(A_{i,j}=0,B_{i,j}=0\right)+ \\ &\quad H\left(V_{i,j}\mid U_{i,j},A_{i,j}\neq 0,B_{i,j}=0\right)\cdot p\left(A_{i,j}\neq 0,B_{i,j}=0\right)+ \\ &\quad H\left(V_{i,j}\mid U_{i,j},A_{i,j}=0,B_{i,j}\neq 0\right)\cdot p\left(A_{i,j}=0,B_{i,j}\neq 0\right)+ \\ &\quad H\left(V_{i,j}\mid U_{i,j},A_{i,j}\neq 0,B_{i,j}\neq 0\right)\cdot p\left(A_{i,j}\neq 0,B_{i,j}\neq 0\right). \end{aligned} \tag{6}$$

The DSC encoder selects a proper channel code and channel code rate. The exact context of each bit is provided only to the DSC decoder, which will use the context information and the statistics of each context to set the *a prior* log likelihood (LLR) of each bit for the turbo or LDPC channel decoder. In short, the LLR of coefficient $j$ at bitplane $i$ is set to:

$$LLR(i,j)=\log\left[\frac{p(V_{i,j}=0\mid U_{i,j},A_{i,j},B_{i,j})}{p(V_{i,j}=1\mid U_{i,j},A_{i,j},B_{i,j})}\right], \tag{7}$$

We may then proceeds with turbo or LDPC channel decoding with Belief propagation.

### 3.2. Turbo vs LDPC based distributed source coding

The channel codes usually considered for DSC are the turbo codes [4] and the LDPC codes [5]. Turbo code is usually constructed via the parallel concatenation of two convolutional codes with an interleaver placed in front of one of the convolutional codes. The construction of the turbo code is systematic, and a new turbo code can be readily generated for any input block length.

In comparison, LDPC code is constructed via a graph linking two sets of nodes: the variable nodes (holding the information bits) and the source nodes (holding the parity bits). The two sets are connected through an edge interleaver. A good LDPC code needs an edge interleaver that is free of short cycles (at lease length-2 and length-4 cycles), and the removal of cycles is the primary computation complexity in LDPC code construction and is very expensive. Therefore, LDPC code needs to be pre-designed with input block length. Moreover, when the channel error rate becomes low, the removal of short cycles becomes difficult to perform. We can only generate successful interleavers at code rates above 0.1.

We compare the BSC channel code performance of the turbo and LDPC code, and show the result in Table 3. In the experiment, we use a simulated BSC channel with error rate $R_i$ being 0.125, 0.25 and 0.5bpp, respectively. We then gradually increase the channel code rate until all errors can be corrected. We record the gap between the channel code rate and the actual channel error rate for turbo and LDPC code, and report the result in Table 3. Larger gap indicates poor channel code performance. We observe that LDPC codes always work better than turbo codes. Moreover, the performance gap between the turbo and LDPC grows larger when the channel error rate becomes high.

Table 3 channel code performance: turbo vs. LDPC.

| Channel error rate | Turbo Loss=$R_2$-$H(V\|U)$ | LDPC Loss=$R_2$-$H(V\|U)$ |
|---|---|---|
| 1/8 | 0.036 | 0.034 |
| 1/4 | 0.060 | 0.038 |
| 1/2 | 0.145 | 0.037 |

Based on the above observations, we use the LDPC code for those bitplanes where the channel error rate is above 0.1, and use the turbo codes for those bitplanes where the channel error rate is below 0.1.

### 3.3. Binary Asymmetric Channel

Existing turbo and LDPC code are designed for BSC channel, where bit 0 occurs in equal probability to bit 1. However, in bitplane coding, only the refinement bit is equally probable. The bit in significant identification is highly skewed towards zero, both for a prior probability of the reference bit U, and for the transfer probability towards the coded bit V. This can be shown in Figure 4. We observe that the LDPC-based DSC does not suffer in performance with regard to the binary asymmetric channel. However, the performance of the turbo-based SWC suffers significantly.

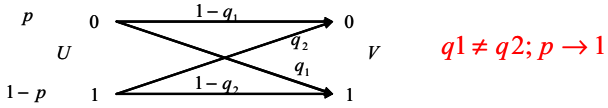


Figure 4. Binary asymmetric channel.

One solution is to re-design the turbo code based on the binary asymmetric channel, as proposed in [9]. However, this involves significant work and is not trivial. In this paper, we adopt a simple solution: we just use a random flipper to transfer the binary asymmetric channel into a BSC channel. We use a pseudo random seed which is synchronized between the encoder and decoder to generate a pseudo random binary stream. This pseudo random stream is then xored onto the $\mathbf{U_i}$ and $\mathbf{V_i}$ before they are DSC encoded. At the time of DSC decoding, the same pseudo random sequence is xored to the reference bitplane to correct the bias. We found the use of random flipper greatly improve the DSC performance of turbo coding. The random flipper is used for all turbo-based DSC in the experimental results.

## 4. EXPERIMENTAL RESULTS

We apply the proposed context DSC for distributed audio coding. We use the experimental setup in Figure 1. The audio source used in the experiment is obtained from the MPEG sound quality assessment material [10]. We assume that the original audio is broadcast by radio, and received by two peers with uncorrelated AWGN noise of 80dB. Each peer then transform the audio via the modified DCT transform (MDCT), and then apply a rate-18 memory-8 TCQ. We want the receiving peers to recover the original audio $\mathbf{X}$ at 80.6dB. Calculation shows that this can be achieved by TCQ quantizing each source with quantization step size 5.6.

Table 4 Separate source coding and distributed source coding: without and with context.

| | No context coding | Context coding | Gain by context coding |
|---|---|---|---|
| Separate source coding | 7.79 | 6.38 | 18.10% |
| Distributed source coding | 6.24 | 4.83 | 22.60% |
| DSC saving | 19.90% | 24.29% | Overall=38% |

We first compare the separate source coding and DSC, both with and without context coding. The results are shown in Table 4. We show that the context modeling is a powerful technology in both source coding and DSC. It achieves a rate reduction of 18% in the source coding, and achieves a rate reduction of 23% in DSC. The performance of DSC without context modeling is only 2% better than the separate source coding with context modeling. However, applying context modeling in DSC, we may achieve an additional saving of 22%, and allow DSC to beat separate source coding significantly.

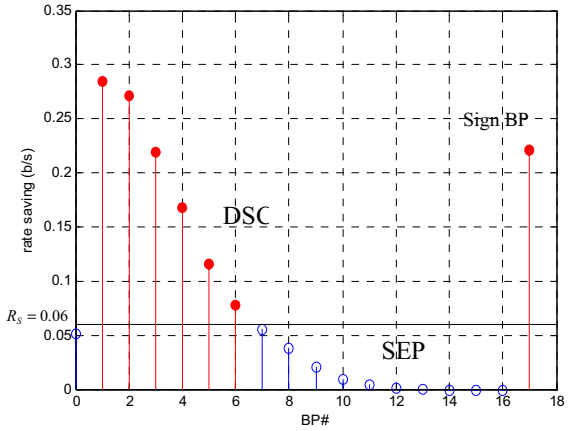


Figure 5 Saving of distributed source coding: bitplane result.

Our entropy analysis also shows that the saving of DSC is highly non-uniform across the bitplanes. We show the rate saving of each bitplane of using DSC vs. separate source coding (both with context modeling) in Figure 5. The horizontal axis is the bitplane index, the vertical index is the DSC rate saving in bit per symbol (bps). We observe that the DSC rate saving is small (below 0.06bps) in bitplanes 0 and 7-16. At most significant bitplanes, the correlation between the bitplanes is strong. However, the entropy of the bitplanes is very low. As a result, the DSC saving is low in bps as well. Because it is difficult to do DSC at low channel error rate, we will simply use separate source coding for those bitplanes. At bitplane 0, the correlation of the two bitplanes is weak. Therefore, DSC does not provide much saving either. As a result, we decide to apply DSC only to bitplanes 1-6 and the sign bitplane. Our decision results in a total rate loss of 0.18bps, which is about 3% of the coding bitrate of separate source coding.

To implement DSC practically, we need to select a proper channel code and code parameter based on the entropy analysis above. Following the discussion in Section 3.2, we apply LDPC code to the bitplanes 1 and 2, where the compound virtual channel error rate is above 0.1 bps. We apply turbo code to the bitplanes 3-6, where the compound virtual channel error rate is below 0.1bps. The DSC code and code parameters used for each bitplane coding of our experiment can be found in Table 5.

Table 5 SWC code selection and code parameter.

| BP# | H(V\|UAB) | Channel Code | Bit rate | Rate loss |
|---|---|---|---|---|
| 17 | 0.307486 | 0.38-LDPC | 0.336453 | 0.028967 |
| 6 | 0.001819 | 140/141-Turbo | 0.007143 | 0.005324 |
| 5 | 0.005725 | 46/47-Turbo | 0.021739 | 0.016014 |
| 4 | 0.015694 | 21/22-Turbo | 0.047621 | 0.031927 |
| 3 | 0.041432 | 12/13-Turbo | 0.083333 | 0.041901 |
| 2 | 0.108174 | 0.85-LDPC | 0.15 | 0.041826 |
| 1 | 0.268990 | 0.7-LDPC | 0.3 | 0.031010 |
| SUM | 0.749320 | - | 0.946289 | 0.196969 |

In Table 5, there is always a performance gap between the channel error rate and the channel code rate. The performance gap is due both to the design of the channel code (only a number of channel code profiles can be pre-designed) and due to the requirement to achieve zero-error in channel decoding in DSC. We observe a further rate loss of 0.2bps, or 3% in practical DSC implementation. Overall, our practical DSC audio coding system achieves a total rate of 5.21bps, which represents an 18% rate saving compared with separate coding of the source. Because one path of DSC is exactly the same as separate source coding, the DSC path actually achieves a rate saving of 36% compared with separate source coding.

## 5. CONCLUSIONS

We introduce context modeling to DSC and develop practical DSC design that switch between turbo and LDPC-based DSC. Context DSC achieves a rate saving of 36% compared with separate source coding.

## REFERENCES


[1] S. Pradhan, J. Kusuma, and K. Ramchandran, "Distributed compression in a dense microsensor network," *IEEE Signal Processing Magazine*, vol. 19, pp. 51-60, March 2002.

[2] Z. Xiong, A. Liveris, and S. Cheng, "Distributed source coding for sensor networks," *IEEE Signal Processing Magazine*, vol. 21, pp. 80-94, September 2004.

[3] D. Slepian and J.K. Wolf, "Noiseless coding of correlated information sources," *IEEE Trans. Info. Theory*, vol. 19, pp. 471-480, July 1973.

[4] C. Berrou and A. Glavieux, "Near optimum error correcting coding and decoding: turbo-codes," *IEEE Trans. Comm.*, vol. 44, pp. 1261-1271, October 1996.

[5] R. Gallager, *Low Density Parity Check Codes*, MIT Press, 1963.

[6] A. Aaron and B. Girod, "Compression with side information using turbo codes," *Proc. DCC'02*, Snowbird, UT, April 2002.

[7] A. Liveris, Z. Xiong and C. Georghiades, "Compression of binary sources with side information at the decoder using LDPC codes," *IEEE Comm. Letters*, vol. 6, pp. 440-442, October 2002.

[8] J. Rissanen, "A universal data compression system", *IEEE Trans. Info. Theory,* vol. IT-29, pp.656-664, Sept. 1983.

[9] J. Li, Z. Tu, and R.S. Blum, "Slepian-Wolf Coding for Nonuniform Sources Using Turbo Codes," *Proc. DCC'04*, Snowbird, UT, March 2004.

[10] "Sound quality assessment material recordings for subjective tests", http://www.tnt.uni-hannover.de/project/mpeg/audio/sqam/.