AN EFFICIENT FINE-GRAIN SCALABLE COMPRESSION SCHEME FOR SPARSE DATA

Stefan Strahl[*] *and Alfred Mertins*

Signal Processing Group
Department of Physics, University of Oldenburg
26111 Oldenburg, Germany

alfred.mertins@uni-oldenburg.de
stefan.strahl@mail.uni-oldenburg.de

ABSTRACT

A fine-grain scalable and efficient compression scheme for sparse data based on adaptive significance-trees is presented. Common approaches for 2-D image compression like EZW (embedded wavelet zero tree) and SPIHT (set partitioning in hierarchical trees) use a fixed significance-tree that captures well the inter- and intraband correlations of wavelet coefficients. For most 1-D signals like audio, such rigid coefficient correlations are not present. We address this problem by dynamically selecting an optimal significance-tree for the actual data frame from a given set of possible trees. Experimental results on sparse representations of audio signals are given, showing that this coding scheme outperforms single-type tree coding schemes and performs comparable to the MPEG AAC coder while additionally achieving fine-grain scalability.

1. INTRODUCTION

Recent advances in sparse signal representation ([1, 2, 3]) have increased the interest to apply these methods on audio data ([4, 5]) and led to the demand for an efficient compression scheme of sparse audio representations. Moreover, the increase of heterogeneous networks like the Internet introduced problems such as bitrate fluctuation, different target channel capacities or storage costs for multi-bitrate files. Storing the data in an embedded manner using significance-trees can address this issues in a generic manner.

Bitplane coding and significance-trees have been successfully applied to image coding ([6],[7]). Such coding schemes successfully capture the structure of the wavelet-based image representation, making very efficient sorting passes and a low number of sorting bits possible. Such natural rigid correlations cannot be found in audio signal representations like e.g. the MDCT transform, necessitating the derivation of optimal significance-trees in a data dependent manner.

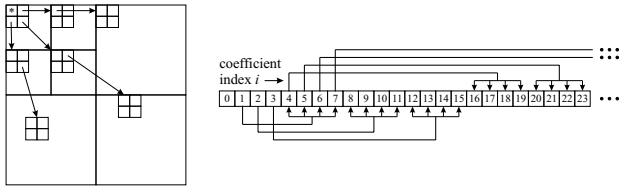
How to generate these significance trees capturing the variant spectral distribution of audio data and the principle of our progressive compression scheme using these significance-trees, referred to as significance tree coding (STC), are discussed in Section 2. In Section 3, we present experimental results on sparse audio representations including subjective listening tests.

[*] This work was partly funded by the DFG through the International Graduate School for Neurosensory Science and Systems at the University of Oldenburg

2. BASIC CONCEPTS

2.1. Significance Trees

Significance-tree coding algorithms like EZW [6] or SPIHT [7] exploit the fact that it can be beneficial, especially for sparse data, to describe significant coefficients of a bitplane via their position and value information instead of transmitting all values one by one. These spatial orientation trees can be mathematically represented using parent-children coefficient coordinate relationships. Fig. 1a shows the case of image compression, were the offspring $O(i,j)$ of the wavelet parent coefficients at position (i,j), except for the highest and lowest pyramid level, have been defined as $O(i,j) = \{(2i,2j),(2i,2j+1),(2i+1,2j),(2i+1,2j+1)\}$. Due to the fact that the 2-dimensional wavelet transformation has a typical coefficient inter- and intra-band correlation [8], this rigid tree structure can capture the correlation with a reasonable computational complexity, giving an efficient compression scheme.



(a) (b)

Figure 1: Parent-offspring dependencies in SPIHT with different styles. (a) 2-D tree. (b) 1-D tree following the offspring rule $O(i) = iN + \{0, N-1\}$.

For 1-dimensional signals like audio data, the problem of selecting the optimal tree structures remains unsolved despite considerable efforts. Most existing algorithms use a single type of tree as shown in Fig. 1b with the fixed parent-children relationship $O(i) = iN + \{0, 1, \cdots, N-1\}$ for different positive integers N. For the MDCT transform, $N = 4$ was adopted in [9, 10, 11, 12] and the wavelet packet transform was encoded using $N = 2$ in [13, 14]. This type of tree will be referenced in the following as SPIHT-style significance trees.

2.2. Bitplane coding using Significance-Trees

The set of M transform coefficients to be encoded for an audio frame is denoted by the vector $\mathcal{X} = (X_1, X_2, \ldots, X_M)$, and the

according coordinates set is denoted by $\mathcal{M} = (1, 2, \cdots, M)$. The algorithm starts with the most significant bitplane n_{max}, which can be easily computed with $n_{max} = \lfloor \log_2(\max_{i \in \mathcal{M}}\{| X_i |\}) \rfloor$. A coefficient X_i can then be expressed as

$$X_i = s \sum_{k=n_{min}}^{n_{max}} b_{i,k} 2^k$$

with $b_{i,k} \in \{0, 1\}$ and $s \in \{\pm 1\}$ being the sign. If X_i is an integer value, then $n_{min} = 0$. To encode real-valued coefficients, n_{min} can be negative.

During the bitplane-coding process, all bitplanes $n \leq n_{max}$ are processed iteratively (i.e., the bits $b_{i,n}$, $i = 1, 2, \ldots, M$ are transmitted) in so-called sorting and refinement passes [7]. In a sorting pass, all coefficients that become significant with respect to the actual bitplane n are found by employing tests on the coefficient absolute values, and these test results are written to the output bitstream. For coefficients that are found to be significant, also a sign bit is transmitted. During the refinement passes, the lower bitplanes of already identified significant coefficients are transmitted.

The sequence of the coefficient sorting is defined by the significance-tree so that all elements in the coefficient set $\mathcal{X}$ are uniquely mapped into nodes in the trees. Each significance tree $\mathcal{T}$ is composed of several nodes that link coefficient coordinates i (position information) of scalars X_i in a hierarchical manner. A tree $\mathcal{T}$ is said to be significant with respect to bitplane n if any scalar inside the tree is significant, that is, if the magnitude of at least one coefficient in the set is larger or equal to 2^n. The pseudocode of the sorting pass is as follows:

TreeSignificance (current tree $\mathcal{T}$, current threshold 2^n)

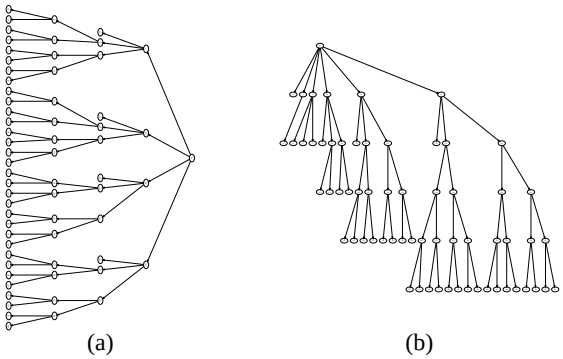
- *If $\mathcal{T}$ is insignificant with respect to 2^n, emit '0' and return;*
- *If $\mathcal{T}$ is significant with respect to 2^n, emit '1';*
- *If root node $N(\mathcal{T})$ is significant with respect to 2^n, emit '1', otherwise emit '0';*
- *Call TreeSignificance() for each subtree with root node as offspring of $N(\mathcal{T})$ with threshold 2^n;*
- *Return;*

2.3. Proposed Adaptive Significance-Tree Selection

The SPIHT-style significance trees proposed for one-dimensional data so far are rather arbitrary. They are simply derived by projecting the known 2-D trees into the vector notation of 1-D structures. To establish better tree structures and in the case of audio data to capture the dynamically variant spectral behavior, we predefine a set of significance-trees and dynamically select the locally optimal ones for each data frame.

For tree construction, in general, it is important to recall that trees should be built in such a way that the coefficients that are most likely to be large in magnitude are located close to the roots of the trees, whereas the small coefficients should be located at the outer leaves. The larger the (sub)-trees that contain small coefficients are, the more efficient the coding will be. In contrast to [15] we used non-complete significance trees by placing remaining nodes at the last treelevel.

In this paper we design the set of μ possible significance-trees by constructing these trees out of m subtrees with different roots and different sorting orders. The coding cost to encode the tree selection information is $log_2(\mu)$ bits per frame. We considered $m = 8$ with equally sized subtrees and $m = 10$ with logarithmically sized subtrees. See Fig. 2 for an illustration of the trees. Each subtree was selected from four different types of trees (ascending, descending, concave oder convex) yielding $\mu = 65.536$ possible trees (tree selection needs 16bit per frame) for the equally sized and $\mu = 1.048.576$ (bit cost of 20bit per frame) for the logarithmically sized subtrees.



(a) (b)

Figure 2: Examples of possible significance-trees with treeorder $N = 2$ and framelength $M = 64$ (a) $m = 4$ (equally sized trees). (b) $m = 6$ (log-sized trees).

For a given data frame to be encoded, we select the tree that allows us to encode the largest number of high-magnitude coefficients within the first ν tree levels. In the experiments, ν was set to 3.

2.4. STC Algorithm

Let us assume that a set of optimal local significance trees for transmitting a coefficient set $\mathcal{X}$ has been found, for example, through testing the efficiencies of various possible trees as mentioned above. The compression scheme then operates as follows: Iteratively, all bitplanes $n = n_{max}, n_{max} - 1, n_{max} - 2, \ldots, n_{min}$ are processed in sorting and refinement passes. In a sorting pass, all coefficients that become for the first time significant (i.e., their magnitude exceeds the current threshold 2^n) are logged to a list of significant coefficients (LSC) and their signs are encoded. This means, at any point in the encoding process, the LSC contains the coordinates of all coefficients that have been found to exceed the current test threshold of 2^n. When all significant coefficients with respect to the current threshold 2^n have been identified and their coordinates have been moved to the LSC, the refinement pass stores the bitplane information for the significant coefficients by processing the LSC, except for the coefficients that were included in the last sorting pass. The overall algorithm is as follows.

STC Algorithm:

1. *Tree Generation:* select one of the μ possible significance-trees, containing m local subtrees;
2. *Initialization:* output $n = \lfloor \log_2(\max_{i \in \mathcal{M}}\{| X_i |\}) \rfloor$; output selected significance-tree; sequentially do: set LSC (list of significant coefficients) as an empty list.

3. *Sorting Pass:* sequentially call *TreeSignificance*, move all significant coefficients into the according LSC, output their signs.
4. *Refinement Pass:* sequentially, for each coefficient in according LSCs, except those included in the last sorting pass, output the n^{th} most significant bit of X_i.
5. *Quantization-Step Update:* decrement n by 1 and go to Step 3.

The process is repeated until the desired bit budget is achieved, or, in case of lossless compression, all bits in all coefficients have been encoded.

3. EXPERIMENTAL RESULTS

3.1. Comparison of compression schemes on sparse audio data

In this section, we compare the performance of run-length coding, Huffman coding, arithmetic coding and adaptively selected and fixed significance trees on sparse audio representations. The parameters for the Huffman coding were set to 8 levels of allowed splitting. For our STC algorithm the number of possible trees was set to $\mu = 65.536$ (equally sized) and $\mu = 1.048.576$ (logarithmically sized), respectively, as described in Section 2.3.

The audio signal was selected as the `cha2.wav` file [16] (mono, 16 bits, 48 kHz) and the bitrate was set to $R = 96$ kbps. To obtain a sparse signal representation of the audio signal a MDCT transform with a framesize $M = 1024$ was used. The MDCT of audio signals already results in a sparse signal representation ([17]) but to increase the sparseness we also used the Basis Pursuit algorithm ([1]) with an overcomplete MDCT-Basis, where the subband signals were oversampled by a factor of two. The frame bit budget R_f was computed as $R_f = \lfloor R \cdot M/Fs \rfloor$ where Fs is the sampling rate in Hz, yielding $R_f = 2048$ bits per frame for 96 kbps. For the compression schemes to achieve the desired bitrate, a linear quantization of the MDCT coefficients has been applied. The treeorder of the significance trees has been set to $N = 4$. As a performance measure, the frame-wise signal-to-noise ratio (SNR) was used, which was computed as the ratio of a frame's energy, divided by the energy of the reconstruction error in the frame. For the two scenarios, we obtained the results listed in Table 1.

Table 1: Average frame-wise SNRs in dB for the `cha2.wav` signal coded at 96 kbps, using different algorithms.

scenario (segSNR)	MDCT	Basis Pursuit overMDCT
RLE	4.94	13.74
Huffman	27.01	25.12
Arith	28.91	26.34
SPIHT	32.99	30.19
STC-lin	34.27	31.47
STC-log	34.56	31.51

From Table 1 it can be seen that an adaptive significance-tree selection benefits from the variant spectral distribution of audio data compared to a fixed significance tree. The representation using critical sampling achieves better results compared to the oversampled case as here the double amount of coefficients needs to be encoded.

3.2. Combination with the MPEG AAC Standard

In this experiment, we use the state-of-the-art MPEG AAC compression scheme and combine it with our STC algorithm in order to achieve progressive coding. For this, we keep the AAC scheme unchanged up to the point where Huffman coding is employed, then apply the STC algorithm to carry out the compression of the quantizer indices. The quantizer indices form a sparse representation of the audio signal. In the experiment, the reference software of [18] was used.

The compression of quantizer indices can either be lossless or lossy, depending on the number of bits transmitted. On the decoder side, the received quantizer indices (either exact values or approximations, depending on the bitrate) are injected into the standard AAC decoder. All other side information is transmitted as produced by the AAC coder.

Note that the results for the AAC coder were produced by encoding the signal individually for each bitrate. For STC, the encoding was done once at 64 kbps, and then lower rates were realized by truncating the frame-wise embedded bitstream produced by the STC algorithm.

To measure the subjective quality, we carried out listening tests comparing the STC-scheme with the MPEG2-AAC standard and the MPEG-4-AAC-BSAC standard, which is currently the only standardized fine-grain progressive audio compression scheme. Twenty test persons for the scenario with eight equally sized subtrees per frame using signals from the sound quality access material (SQAM) [19] and from the 1990 MPEG evaluation [16] have been used.

The measurement procedure was set up according to the ITU recommendation BS.1116-1 [20]. The quality ratings between one (very annoying) and five (indistinguishable from the original) were translated into the subjective difference grade, which is the difference between the rating for the encoded test item and the hidden reference and ranges from zero (equal quality) down to -4 (the lowest grade). The results for three different test signals are depicted in Fig. 3. As one can see, the performance of STC is almost equal to the AAC performance, and it is significantly better than the BSAC one.

4. CONCLUSIONS

The fine-grain scalable sparse audio representation compression problem has been addressed in this study. While in almost all existing algorithms, a single type of significance-tree has been adopted for sorting significant coefficients and transmitting position information, we have proposed a novel adaptive significance-tree technique. Such a tree is generated dynamically to suit variant spectral behavior from frame to frame. Based on the dynamic tree selection, a compression scheme has been proposed which provides both high compression quality and fine-grain bitrate scalability. Experimental results clearly demonstrate the following properties: the method outperforms the existing SPIHT-like algorithms and yields competitive quality as the nonscalable AAC audio compression scheme, yet with fine scalability of one-bit granularity per frame.

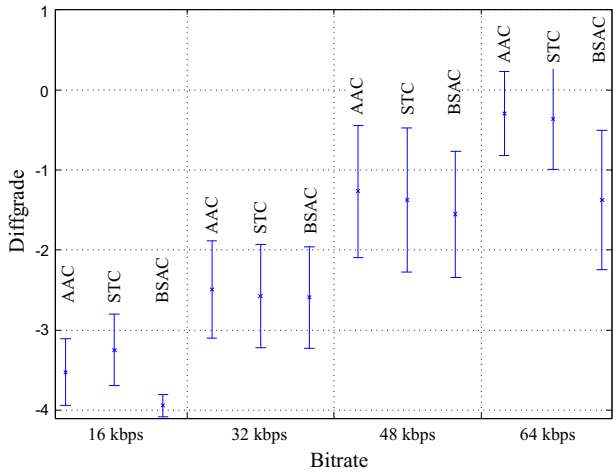


Figure 3: *Subjective difference grades for different codecs at bitrates between 16 and 64 kbps for one mono channel.*

5. REFERENCES

[1] S. S. Chen, D. L. Donoho, and M. A. Saunders, "Atomic decomposition by basis pursuit," *SIAM Journal on Scientific Computing*, vol. 20, no. 1, pp. 33–61, 1999. [Online]. Available: citeseer.ist.psu.edu/chen98atomic.html

[2] M. S. Lewicki and T. J. Sejnowski, "Learning overcomplete representations," *Neural Computation*, vol. 12, no. 2, pp. 337–365, 2000. [Online]. Available: citeseer.ist.psu.edu/lewicki98learning.html

[3] J. Fuchs, "Sparsity and uniqueness for some specific underdetermined linear systems," in *Proc. IEEE ICASSP 2005, Philadelphia, USA*, March 2005.

[4] M. Davies and L. Daudet, "Sparse audio representations using the mclt," *in press*, 2005.

[5] R. Gribonval, "Sparse decomposition of stereo signals with matching pursuit and application to blind separation of more than two sources from a stereo mixture," in *ICASSP02, Orlando, Florida, USA*, May 2002.

[6] J. M. Shapiro, "Embedded image coding using zerotrees of wavelet coefficients," *IEEE Trans. on Signal Processing*, vol. 41, no. 12, pp. 3445–3462, 1993.

[7] A. Said and W. A. Pearlman, "A new, fast and efficient image codec based on set partitioning in hierarchical trees," *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 6, no. 3, pp. 243–250, 1996.

[8] Z. Liu and L. J. Karam, "Quantifying the intra and inter subband correlations in the zerotree-based wavelet image coders," in *Conf. Record of the 36th Asilomar Conf. on Signals, Systems and Computers*, Sep. 2002, pp. 1730–1734.

[9] C. Dunn, "Efficient audio coding with fine-grain scalability," in *AES 111th Convention*. NY, USA: preprint 5492, Sep. 2001.

[10] M. Raad, A. Mertins, and R. Burnett, "Audio coding based on the modulated lapped transform (MLT) and set partitioning in hierarchical trees," in *Prof. 6th World Multiconference on Systemics, Cybernetics and Informatics*, Orlando, USA, Jul. 2002, pp. 303–306.

[11] M. Raad and A. Mertins, "From lossy to lossless audio coding using SPIHT," in *Proc. of the 5th Int. Conf. on Digital Audio Effects*, Hamburg, Germany, Sep. 2002, pp. 245–250.

[12] M. Raad, A. Mertins, and R. Burnett, "Scalable to lossless audio compression based on perceptual set partitioning in hierarchical trees (PSPIHT)," in *Proc. Int. Conf. on Acoustics, Speech, and Signal Processing*, Apr. 2003, pp. V624–627.

[13] Z. Lu and W. A. Pearlman, "An efficient, low-complexity audio coder delivering multiple levels of quality for interactive applications," in *Proc. IEEE Signal Processing Society Workshop on Multimedia Signal Processing*, Dec. 1998, pp. 529–534.

[14] ——, "High quality scalable stereo audio coding," 1999. [Online]. Available: http://www.cipr.rpi.edu/ pearlman/papers/scal_audio.ps.gz

[15] S. S. H. Zhou, A. Mertins, "An efficient, fine-grain scalable audio compression scheme," in *Proc. AES 118th Convention, Barcelona, Spain*, May 2005.

[16] ISO/MPEG, "Audio test report. ISO/IEC/JTC 1/SC 2/WG 11 MPEG MPEG90/N0030," *International Organization for Standardization*, 1990.

[17] M. Davies and N. Mitianoudis, "Simple mixture model for sparse overcomplete ica," in *IEE Proc.-Vis. Image Signal Process., Vol. 151, No. 1*, February 2004.

[18] "Mpeg-4 audio reference software." [Online]. Available: http://www.iso.ch/iso/en/ittf/PubliclyAvailableStandards/ISO_IEC_14496-5_2001_Software_Reference/

[19] "Sound quality assessment material." [Online]. Available: http://sound.media.mit.edu/mpeg4/audio/sqam/

[20] ITU-R Recommendation BS.1116-1, "Methods for the subjective assessment of small impairments in audio systems including multichannel sound systems," *International Telecommunication Union, Geneva*, Dec. 1997.