AN ADAPTIVE TREE-BASED PROGRESSIVE AUDIO COMPRESSION SCHEME

Stefan Strahl, Huan Zhou, 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 audio compression scheme 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 intra-band correlations of wavelet coefficients. For 1-D audio signals, such rigid coefficient correlations are not present. We address this problem by dynamically selecting an optimal significance-tree for the actual audio frame from a given set of possible trees. Experimental results 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 wireless audio streaming ([1],[2]) and 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 can address this issue in a generic manner.

Bitplane coding and significance-trees have been successfully applied to image coding ([3],[4]). 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 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, called combined significance-tree quantization (CSTQ) using these significance-trees, are discussed in Section 2. In Section 3, we present experimental results on audio compression including subjective listening tests.

This work was partly funded by the German Science Foundation (DFG) through the International Graduate School for Neurosensory Science and Systems

2. BASIC CONCEPTS

2.1. Significance trees

Significance-tree coding algorithms like EZW [3] or SPIHT [4] exploit the fact that it can be beneficial 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. Figure 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 [5], this rigid tree structure can capture the correlation with a reasonable computational complexity, giving an efficient compression scheme.

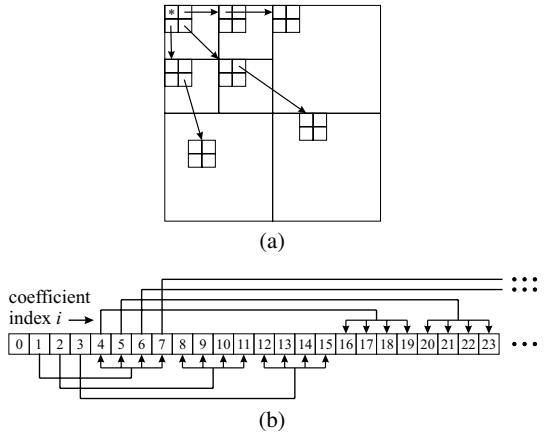

Figure 1: Parent-offspring dependencies in SPIHT with different styles. (a) 2-D tree. (b) 1-D tree following the offspring rule $O(i) = iN + \{0, N-1\}$.

For 1-dimensional audio signals, 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 Figure 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 [6, 7, 8, 9] and the

wavelet packet transform was encoded using $N = 2$ in [10, 11]. 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 [4]. 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 than 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 audio coding 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 to capture the dynamically variant spectral behavior of audio signals, we predefine a set of significance-trees and dynamically select the locally optimal ones for each audio 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 this paper we design the set of μ possible significance trees by partitioning an audio frame into m different segments for which separate subtrees with individually chosen sorting orders are built. Each subtree is selected from four different types of trees that are designed for ascending, descending, concave, and convex coefficient-magnitude behavior within a segment. For example, a tree for a descending model is constructed under the assumption that the coefficients in a segment, denoted by x_i, $i = 1, 2, \ldots, I$ satisfy the condition $|x_i| \geq |x_{i+1}|$. The structure of a subtree for a descending model with treeorder $N = 2$ (number of offspring) can then take on the form illustrated in Figure 2. The trees for the other models can be easily derived from the one for the descending model by rearranging the coefficient indices.

Note that the above mentioned condition $|x_i| \geq |x_{i+1}|$ is just the assumption behind the subtree construction. The actual coefficients in a segment will, of course, not follow such a strict rule. For a given frame segment to be encoded, we select the model 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.

In Section 3, we consider $m = 8$ with equally sized subtrees and $m = 10$ with logarithmically sized subtrees. For logarithmically sized subtrees, the number of coefficients in a segment doubles from segment to segment. With m segments and four trees to choose from in each segment, this yields $\mu = 65.536$ possible trees (tree selection needs 16 bits per frame) for the equally sized and $\mu = 1.048.576$ (bit cost of 20 bits per frame) for the logarithmically sized subtrees.

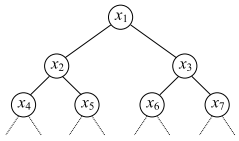

Figure 2: Significance tree for descending model and $N = 2$.

2.4. CSTQ 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.

CSTQ 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 significance-tree models

In this section, we compare the performance of adaptively selected and fixed significance trees. The number of possible trees for our algorithm 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 [12] (mono, 16 bits, 48 kHz) and the bitrate was set to $R = 96$ kbps. An MDCT filterbank was used to remove the signal redundancy and the framesize was set to $M = 1024$. 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. The treeorder of the significance trees has been set to $N = 4$. As a quality 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. The two scenarios gave 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	SPIHT	CSTQ linear spaced	CSTQ log-spaced
segSNR	32.99	34.27	34.56

From Table 1 it can be seen that the CSTQ algorithm, which uses the proposed adaptive significance-tree selection, gives better results than the SPIHT scheme with a fixed, signal independent tree. Moreover, it can be seen that the use of a logarithmic spacing, similar to the one that can be found in the human auditory system, is a good strategy to exploit the structure of audio signals.

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 CSTQ 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 CSTQ algorithm to realize the compression of the quantizer indices. In all experiments, the reference software of [13] 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.

Table 2 shows the average segmental SNRs for the algorithms at different bitrates, using signals from the sound quality access material (SQAM) [14] and from the 1990 MPEG evaluation [12]. Note that the results for the AAC coder were produced by encoding the signal individually for each bitrate. For CSTQ, the encoding was done once at 64 kbps, and then lower rates were realized by truncating the frame-wise embedded bitstream produced by the CSTQ algorithm. As the results in Table 2 show, the SNR for CSTQ is slightly lower at the highest bitrate, but it is better for all lower bitrates. A similar behavior could be found for other audio material as well. This could be explained by to the fact that at 64 kbps, not all frames could be compressed by the CSTQ scheme in a lossless manner within the given bit budget. At lower rates, however, CSTQ has the advantage that it can exactly meet the target bitrate without the need of including any padding bits, which are quite common in the AAC bitstream produced by the reference software.

Table 2: Average segmental SNRs in dB for different signals, bitrates and algorithms

audio file	Time (min)	Bitrate (kbps)	AAC	AAC BSAC	AAC CSTQ linearly spaced $N = 2$	AAC CSTQ log. spaced $N = 2$
Tracy Chapman [12]	0:37	16	10.46	1.26	7.65	8.02
		24	12.90	8.25	9.57	10.32
		32	14.19	12.08	13.63	13.87
		40	15.04	14.04	14.89	14.96
		48	15.59	14.94	15.43	15.49
		56	16.09	15.51	16.01	16.03
		64	16.47	15.54	16.43	16.44
female English speech [14]	0:21	16	7.65	5.59	9.74	9.98
		24	10.48	10.03	12.51	13.30
		32	12.54	12.66	15.50	15.74
		40	13.70	15.53	18.07	18.12
		48	15.28	16.97	19.36	19.41
		56	16.75	17.05	19.89	19.91
		64	19.98	17.07	19.98	19.98
quartet [14]	0:28	16	7.42	6.03	8.51	8.95
		24	9.59	9.48	10.67	11.03
		32	11.32	11.43	13.37	13.51
		40	12.73	13.89	15.23	15.30
		48	14.29	14.77	16.32	16.36
		56	15.82	14.95	16.84	16.86
		64	17.05	14.95	17.03	17.03

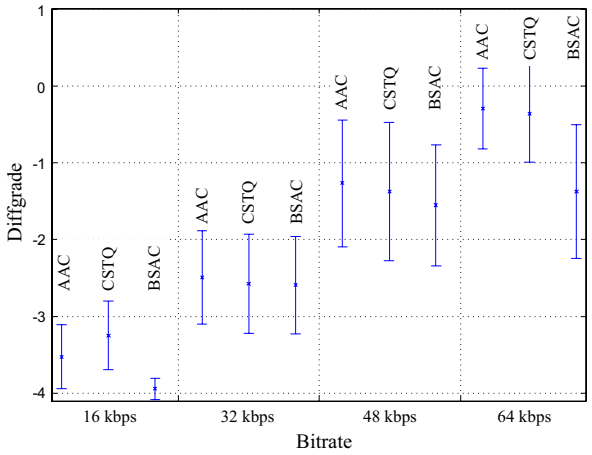

Figure 3: Subjective difference grades for different codecs at bitrates between 16 and 64 kbps for one mono channel.

3.3. Subjective listening tests

In order to see whether the objective results based on the segmental SNR translate into similar subjective quality impressions, we carried out listening tests with twenty test persons for the scenario with eight equally sized subtrees per frame. In these tests, the CSTQ-scheme was compared with the MPEG2-AAC standard and the MPEG-4-AAC-BSAC standard, which is currently the only standardized fine-grain progressive audio compression scheme. Also for MPEG-4-AAC-BSAC, the reference software from [13] was used. The measurement procedure was set up according to the ITU recommendation BS.1116-1 [15]. 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 Figure 3. As one can see, the performance of CSTQ is almost equal to the AAC performance, and it is significantly better than the BSAC one.

4. CONCLUSIONS

The fine-grain scalable audio signal 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. It could be shown that a logarithmic tree size scaling captures better the harmonic structure of an audio signal. Based on the dynamic tree selection, a compression scheme called CSTQ 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.

5. REFERENCES

[1] Bluetooth Audio Video Working Group, "Advanced Audio Distribution Profile Specification (A2DP)," *Bluetooth Special Interest Group*, 2003.

[2] Apple Inc., "AirTunes," 2004. [Online]. Available: http://www.apple.com/airportexpress/airtunes.html

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

[4] 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.

[5] 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.

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

[7] 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.

[8] 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.

[9] 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.

[10] 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.

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

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

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

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

[15] 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.