Audio Engineering Society

Convention Paper

Presented at the 118th Convention
2005 May 28–31 Barcelona, Spain

An Efficient, Fine-Grain Scalable Audio Compression Scheme

Huan Zhou[1], Alfred Mertins[1], and Stefan Strahl[1]

[1]*Signal Processing Group, Department of Physics, University of Oldenburg, 26111 Oldenburg, Germany*

Correspondence should be addressed to Alfred Mertins-Author (`alfred.mertins@uni-oldenburg.de`)

ABSTRACT

To address the fine-grain scalable audio compression issue, a novel combined significance tree technique is proposed for high compression efficiency. The core idea is to dynamically adopt a set of locally optimal significance trees, instead of following the common approach of using a single type of tree. Two different encoding strategies are proposed: the spectral coefficients can be encoded either in a threshold-by-threshold manner or in a segment-by-segment manner. The former yields rate and fidelity scalability, and the latter yields bandwidth scalability. Experimental results show that our proposed scheme significantly outperforms the existing schemes using single-type trees and performs comparably with the MPEG AAC coder while achieving fine-grain scalability.

1. INTRODUCTION

The growth of the Internet together with media streaming over wireless links has created a demand for high-quality streamed audio content. Audio coding with fine-grain bitrate scalability and progressive transmission allows real-time streaming with low buffer delay and uninterrupted service in the presence of channel congestion, and yields the most efficient use of available channel bandwidth. Among the popular audio codecs, the most classical ones do not offer progressive transmission (e.g., MP3, REAL, WMA, AAC) or provide only coarse layers (MPEG-4 AAC). The state-of-the-art scalable audio codec is the MPEG AAC-BSAC, which yields fine scalability (1kbit/s) at the cost of cumbersome layer structures and numerous arithmetic coding models.

In general, the scalability in scalable codecs is obtained at the price of degradation in performance relative to fixed-bitrate versions with the same coding technique. And in general, the finer the granularity is, the higher the loss [1]. Therefore, it is very desirable and attractive to construct a scalable codec with both fine scalable granularity and competitive efficiency. Recently, several works addressed the issue [2, 3, 4, 5, 6, 7, 8, 9, 10] by proposing fine-grain scalable audio compression schemes using the techniques of both ordered bitplane coding and tree-based significance mapping. The basic idea herein is to encode the transformed coefficients by frames. In each frame, all coefficients are in their binary representation. Bitplane encoding (or bit-slicing) technique is to "slice" the coefficients one bit at a time, starting from the most significant bit (MSB) with decreasing order. In each bitplane, a tree-based significance mapping technique is exploited to quickly locate those significant coefficients (whose MSBs locate on the current bitplane) and efficiently convey their position information so that the decoder can accurately relocate them.

With the bitplane coding technique, a coarse representation of the largest (and most significant) coefficients is provided by the top slice, more accurate representations of the most-significant coefficients, and coarse approximations of the next most significant ones are provided by subsequent slices, and so on. Thus, the bitplane coding technique inherently provides rate/fidelity scalability.

The idea of significance-tree coding has originated from image compression and has produced impressive advances in wavelet-based image compression. A well known example is the set partitioning in hierarchical trees (SPIHT) algorithm, proposed by Said and Pearlman [11]. In this scheme, a known coefficient significance/magnitude distribution is assumed in the form of spatial orientation trees, where the coefficients closer to the roots of the trees are expected to be more significant (i.e., larger in magnitude) than those at the leaves. Then, at each bitplane level, the algorithm uses a sorting pass (involving significance tests and recursive partitioning of significant trees) in order to identify the coefficients that are significant with respect to the current bitplane, and then uses a refinement pass to output the current bitplane values for those coefficients that have become significant in one of the previous bitplanes. Through repeated sorting and refinement passes with decreasing bitplane order, the information on both positions and values of significant coefficients is progressively transmitted.

Mathematically, the spatial orientation trees are represented by parent-children coefficient coordinate relationships. For wavelet-based image compression, the offspring of a parent coefficient at position (i,j) are typically defined as $O(i,j) = \{(2i,2j),(2i,2j+1),(2i+1,2j),(2i+1,2j+1)\}$, which holds for all coefficients, except the ones at the highest and lowest pyramid levels. It has been realized that such a significance-tree definition successfully captures not only the inter-band correlation but also the intra-band correlation [12], which results in very efficient sorting passes, producing a low number of sorting bits. Overall, the SPIHT scheme has fine bitrate/fidelity scalability, state-of-the-art compression performance and reasonable computation complexity.

For audio signal compression, the problem of defining optimal tree structures remains unsolved despite considerable efforts. Almost all existing algorithms adopted a single type of tree with the parent-children relationship $O(i) = iN + \{0,1,\cdots,N-1\}$ for different positive integer N. In particular, $N = 4$ was adopted in [2, 5, 6, 7] for the MDCT transform, and $N = 2$ was used in [3, 4] for the wavelet packet transform. These significance tree choices, in nature, are rather arbitrary. They are following the spatial orientation-tree idea with slight modifications of the original matrix notation of 2-D trees into vector notation of 1-D structures. In the following, this type of significance trees will be referred to as SPIHT-style significance trees.

The main difficulties in addressing efficient significance tree structure for audio compression are:

1. The elegant hierarchical self-similarity properties for a dyadic wavelet transform are not present for 1-D non-wavelet transforms like the MDCT, which is a commonly adopted transform form in audio compression.

2. The statistical properties of audio signals might change drastically over time. For the same audio signal source, at different time instances, its spectral magnitude distribution varies dynamically, and a tree that is good for one time instance can be poor for another one.

All these factors make the issue of finding a universal yet efficient significance tree for non-wavelet transformed audio signal compression more complicated.

To address this problem, we propose an adaptive, combined tree-based significance mapping technique, and based on it, we develop a novel, scalable compression scheme, called combined significance tree quantization

(CSTQ). Unlike in sorting with single-type significance trees, we adaptively generate different significance trees in our proposed scheme, depending on the actual signal. Thus the scheme has the natural features of flexibility, adaptivity and fine-grain scalability.

Further flexibility is offered in the CSTQ scheme by two different encoding manners: either sorting the coefficients threshold-by-threshold (CSTQ-T algorithm) over all segments, or by sorting them in a segment-by-segment manner (CSTQ-S algorithm). Encoding threshold-by-threshold yields rate and fidelity scalability, and encoding segment-by-segment yields bandwidth scalability. The CSTQ-S algorithm is more complex than CSTQ-T, because it requires an optimal distribution of the available bits to the segments. In this paper, we confine ourselves to describing the CSTQ-T algorithm.

2. TREE-BASED SIGNIFICANCE MAPPING

2.1. Basic Concept of Significance Tree

Let the vector $\mathscr{X} = (X_1, X_2, \cdots, X_M)$ denote a set of M transform coefficients to be encoded, with according coordinates set $\mathscr{M} = (1, 2, \cdots, M)$. For the coefficient set, the most significant bitplane (MSB) n_{max}, is naturally decided so that $2^{n_{max}} > \max\limits_{i \in \mathscr{M}}(\mid X_i/2 \mid)$. The bitplane n_{max} will be the initial bitplane in the algorithm. Given any bitplane n with $n \leq n_{max}$, all coefficients that become significant with respect to bitplane n are found in a sorting pass that employs tests on coefficient magnitudes.

With the aid of the introduced significance trees, all elements in the coefficient set $\mathscr{X}$ are uniquely mapped into nodes in trees. Each significance tree $\mathscr{T}$ is composed of several nodes that link coefficient coordinates i (position information) of scalars X_i in an hierarchical manner. In this case, the tree $\mathscr{T}$ 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.

During the sorting pass, significance tests are performed on the basis of trees as follows (pseudocode):

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

Observing the above significance test process, it is clear that choosing different significance-tree structures would definitely lead to different bit costs in the sorting pass, and also to the coding efficiency of the entire coding scheme.

2.2. Generation of Significance Trees

To establish a suitable tree structure, considering dynamically variant spectral behavior, there exists no single tree that captures the significance information of all frames equally well. We therefore aim at dynamically finding the best significance tree for each frame from a given set of possible trees. For this, the coefficient set $\mathscr{X}$ is first divided into m segments, and the magnitude distribution curves in each segment are classified by four description models as illustrated in Fig. 1. For the model selection in each segment, we use the following simple procedure. We first divide a given segment into four, equally sized parts and observe the maximum coefficient magnitudes in these four sub-segments. Let the maxima (from left to right) in the four parts of the ith segment be $\xi_{i,1}$, $\xi_{i,2}$, $\xi_{i,3}$, $\xi_{i,4}$, and let $\xi_{i,max} = \max_{k=1}^{4} \xi_{i,k}$ be the overall maximum. Then, if $\xi_{i,max}$ is equal to $\xi_{i,2}$ or $\xi_{i,3}$ we choose model 1. If $\xi_{i,max} = \xi_{i,1}$ and $\xi_{i,1} \geq 2\xi_{i,4}$, we pick model 4. If $\xi_{i,max} = \xi_{i,4}$ and $\xi_{i,4} \geq 2\xi_{i,1}$, we use model 3. If none of these conditions applies, we use model 2.

Alternative to the maximum coefficient magnitudes, the norms of the subsegments can be used to define $\xi_{i,1}$, $\xi_{i,2}$, $\xi_{i,3}$, $\xi_{i,4}$. This increases the cost for model selection, but yields a slight performance improvement.

For notational convenience, in the following, we use $\mathscr{T}^j$ to denote the local significance tree satisfying magnitude distribution model j. Moreover, it should be noted that, in principle, if the tree $\mathscr{T}^4$ is fixed, the tree $\mathscr{T}^3$ can be derived from it through reversing the coefficient sequence.

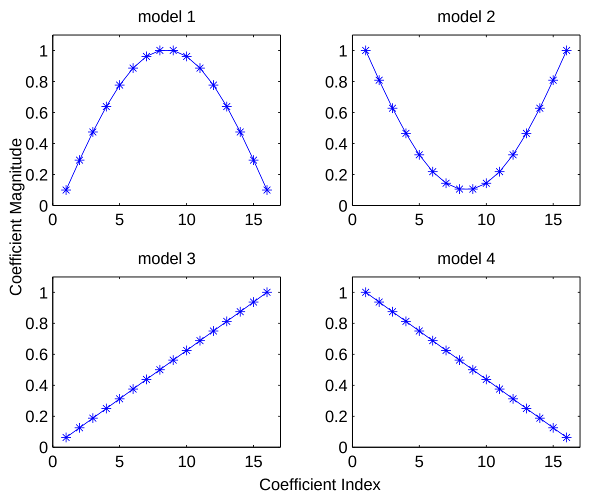

Fig. 1: Approximation magnitude distribution models (model 1,2: symmetric models; model 3: increasing model; model 4: decreasing model).

Similarly, the trees $\mathscr{T}^1$ and $\mathscr{T}^2$ can be obtained from $\mathscr{T}^4$ through reordering the coefficients. Keeping this in mind, our following analysis will only focus on establishing $\mathscr{T}^4$.

Consider a significance tree $\mathscr{T}^4$ (with height of J) that is linked to a length-ω segment of coefficients $\Omega = \{x_1, x_2, \ldots, x_\omega\}$. For the i^{th} subtree with height j, $t_i^{(j)}$, let $C_i^{(j)}$ be its significance mapping cost ('position' bit cost); $P_i^{(j)}$ be the subtree significance probability (including the root node); $n_i^{(j)}$ be its root node; $p_i^{(j)}$ be the root node significance probability; $\{S_{i,j-1}\}$ be the offspring set of the root node; and $s_{i,j-1}$ be the set length. Recalling the pseudocode for tree significance tests, we can derive the significance and bits cost expressions for the subtree $t_i^{(j)}$ as following:

- $j = 1$:

$$\begin{cases} P_i^{(j)} &= 1-(1-p_i^{(j)})\prod_{n\in\{S_{i,0}\}}(1-p_n^0) \\ C_i^{(j)} &= 1+P_i^{(j)}(1+s_{i,0}) \end{cases}$$

- $2 \le j \le J$:

$$\begin{cases} P_i^{(j)} &= 1-(1-p_i^{(j)})\prod_{n\in\{S_{i,j-1}\}}(1-P_n^{(j-1)}) \\ C_i^{(j)} &= 1+P_i^{(j)}(1+\sum_{n\in\{S_{i,j-1}\}}C_n^{(j-1)}) \end{cases}$$

To simplify the problem, in this study, we limit the offspring number of each subtree to a fixed number r (except for trees with highest or lowest height). Then it is easy to get the height J of the entire tree due to the constraint that $(1+r+\cdots+r^J) \le \omega$. So the top height J is the maximum integer solution: $J = \lfloor \log_2((r-1)\omega+1)\cdot(\log_2 r)^{-1} - 1 \rfloor$. The unaccounted $q = \omega - \sum_{i=0}^{J} r^i$ coefficients are assumed as zero-offspring nodes.

To associate the coefficients position information (coordinates) with nodes at different locations in the significance tree, two useful observations are exploited: 1) the significance probability of a higher tree is no less than that of any lower tree. That is, $P_i^{(j)} > P_n^{(j-1)}$ for $1 \le j \le J$ and $n = 1, 2, \ldots, s_{i,j-1}$. 2) for efficient compression, a subtree that is expected to be insignificant should have as many coefficients as possible while a subtree that is expected to be significant should have the lowest possible number of coefficients.

Following these observations, it is natural to decide the root nodes for height J and $J-1$ trees as:

$$n_1^{(J)} = x_1 \tag{1}$$

$$n_{1,\cdots,q+r}^{(J-1)} = [x_{2:q+1}, x_{q+2:q+r+1}] \tag{2}$$

where nodes in $n_{1,\cdots,q+r}^{(J-1)}$ are arranged from left to right. That is, the coefficient with the highest significance probability is chosen as the tree root node; the next $q+r$ most significant coefficients are arranged as the offspring of the root node. If $J = 1$, there are only two levels and all root nodes in height $J-1$ of the tree are zero-offspring nodes, so that the whole tree is generated. Otherwise, if $J > 1$, the first part in the right-hand side (RHS) of (2) corresponds to zero-offspring nodes, and the second part refers to r-offspring nodes.

Now we consider how to arrange the remaining coefficients to all trees with height $0 \le j \le J-2$. Based on the above proposed probability and cost expressions, and with the simplifying assumption of equal costs for encoding subtrees $t_n^{(j-1)}$ for $\forall n \in \{S_{i,j-1}\}$, we have

$$\min(C_i^{(j)}) \Longleftrightarrow \min\left(P_i^{(j)} \sum_{n=1+(i-1)r}^{i\cdot r} C_n^{(j-1)}\right)$$
$$\Longleftrightarrow \max\left((1-p_i^{(j)}) \prod_{n=1+(i-1)r}^{i\cdot r} (1-P_n^{(j-1)})\right).$$

Observing the last expression, we can see that, in order to minimize bit costs, if $p_i^{(j)} > p_{i+1}^{(j)}$, we have $\phi_i > \phi_{i+1}$ with $\phi_i := \prod_{n=1+(i-1)r}^{i\cdot r} P_n^{(j-1)}$. In other words, the bigger the parent is, the bigger offspring are expected. Thus, the significance tree can be easily constructed by arranging the coefficients $x_{q+r+2:w}$ from top to bottom for different heights j with $0 \leq j \leq J-2$ and from left to right for the same height. Together with (1) and (2) for higher tree nodes, the entire significance tree $\mathscr{T}^4$ can be generated quickly. Following this way, all m significance trees are produced, which are concatenated together to form a combined sequence of significance trees for coefficient set $\mathscr{X}$.

3. CSTQ-T ALGORITHM DESCRIPTION

Suppose for a set of coefficients $\mathscr{X}$ that is partitioned into m subsets $\Omega_1, \Omega_2, \ldots, \Omega_m$, the subset significance trees are first generated as described above. The CSTQ-T algorithm compresses the coefficient set $\mathscr{X}$ in the order of threshold-by-threshold, over all segments $\Omega_1, \Omega_2, \ldots, \Omega_m$. The entire algorithm is described as follows:

CSTQ-T Algorithm:

1. *Tree Generation:* Partition the coefficient set X into m segments; sequentially generate local significance trees for the segments;
2. *Initialization:* output $n = \lfloor \log_2(\max_{i\in\mathscr{M}}\{\mid X_i \mid\})\rfloor$; sequentially do: output segment length and significance tree type; 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 2.

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.

Compared to the SPIHT procedure [11], our proposed CSTQ-T has competitive calculational complexity, even with the additional combined significance tree generation process (Step 1). Since our combined significance tree is more accurate and flexible than fixed trees, a higher coding efficiency is expected. This will be confirmed in the experiments in Section 4. The amount of side information required to transmit the significance tree sequence information ($2m$ binary bits) and the segment length information (which could be reduced by uniform division) is relatively low in relation to the number of bits saved due to an enhanced significance tree selection.

4. EXPERIMENTAL RESULTS

Now we apply our proposed CSTQ scheme to the compression of audio signals. First, we compare our scheme with existing algorithms with SPIHT-style significance tree adoption. Then, we combine our method with the state-of-the-art MPEG AAC compression scheme by replacing the Huffman coding stage for quantizer-index compression by the CSTQ algorithm. Both objective and subjective results will be presented.

4.1. Comparison with Previous SPIHT-related Schemes

In the experiment, the audio signal was selected as the `svega.wav` file, a female singing, with 44.1 kHz sampling rate and 16 bits per sample. The audio signal was transformed with the MDCT with a constant frame length of $M = 1024$. The target bitrate for a mono channel was chosen as 96 kbps. Four compression algorithms based on the following tree constructions were applied:

1. Adopting SPIHT-style significance trees on each frame (in short, SPIHT);

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

SPIHT	SPIHT+seg	CSTQ-T	CSTQ-T1
30.76	28.89	31.50	28.72

2. Adopting SPIHT-style significance trees on each segment (in short, SPIHT+seg);
3. Using the proposed algorithm with $m = 8$ segments (in short, CSTQ-T)
4. Using the proposed algorithm for tree construction, but with only one segment per frame (in short, CSTQ-T1)

The frame-wise SNRs for the first 812 consecutive frames containing music are given in Table 1. We can see from these results that, overall, CSTQ-T yields the best performance. We also see that the SPIHT algorithm works better on a frame basis, whereas the CSTQ-T algorithm works best in smaller segment basis. To have a more detailed impression of the performance, Fig. 2 visualizes the frame-wise SNRs for the first 100 frames. We can observe that in almost all frames, CSTQ-T outperforms SPIHT, and it always outperforms SPIHT+seg. This implies that for audio signal compression, due to its variant spectral distribution from frame to frame, dynamic tree allocation in CSTQ-T shows more advantages over the single-type trees in SPIHT. Interestingly, when comparing SPIHT and SPIHT+seg, it turns out that SPIHT+seg works better than SPIHT in a few frames. This phenomenon also suggests some disadvantages of adopting single-type trees on frame-wise coding.

4.2. 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-T algorithm in order to achieve scalable coding. For this, we keep the AAC scheme unchanged up to the point where Huffman coding is employed, then apply CSTQ-T algorithm to realize the compression of quantizer indices. In all experiments, the reference software of [13] was used.

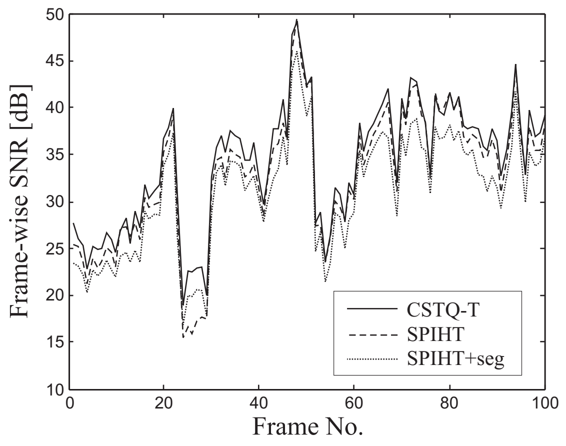

Fig. 2: Frame-wise SNRs for the first 100 frames of the svega signal, coded at 96 kbps.

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 two algorithms at different bitrates, using the quartet signal from the sound quality access material (SQAM), obtained from [14]. Note that the results for the AAC coder were produced by encoding the signal individually for each bitrate. For CSTQ-T, 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-T algorithm. As the results in Table 2 show, the SNR for CSTQ-T 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-T scheme, in a lossless manner within the given bit budget. At lower rates, however, CSTQ-T 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.

In order to see whether the objective results based on the

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

Bitrate	AAC	CSTQ-T
16	7.42	8.51
24	9.59	10.67
32	11.32	13.37
40	12.73	15.23
48	14.29	16.32
56	15.82	16.84
64	17.05	17.03

segmental SNR translate into similar subjective quality impressions, we carried out listening tests with twenty test persons. 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 Fig. 3.

Observing the difference grades in Fig. 3, we see that, overall, the subjective ratings for CSTQ were quite close to those for AAC. Moreover, at the lowest rate of 16 kbps, CSTQ was always rated better than AAC; at the highest rate of 64 kbps, both the AAC and the CSTQ coder achieved close to transparency quality. Further, it is interesting to take a close look at the results for the quartet signal. Comparing the subjective results in Fig. 3 with the objective results listed in Tab. 2 confirms that it is not always the case that an objective advantage will translate into a better subjective quality. The reason for the better subjective performance of AAC at 32 and 48 kbps is likely to be the fact that the AAC coder used individually optimized quantizers (based on the psychoacoustic analysis) for each bitrate, whereas the CSTQ coder used the quantizers that were optimized for 64 kbps and then dropped the lower bitplanes in order to meet the desired target bitrates.

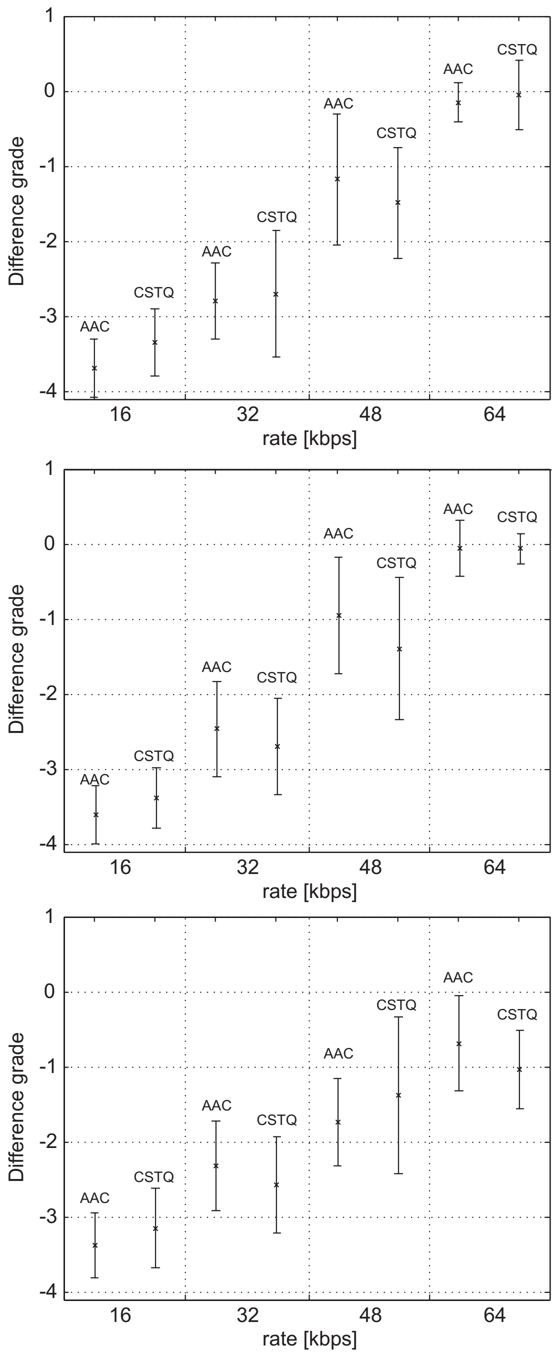

Fig. 3: Subjective difference grades. Top: Tracy Chapman. Middle: Quartet. Bottom: Female English speech.

5. 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 combined 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 called CSTQ-T has been fully developed, which provides both high compression quality and fine-grain bitrate scalability. Further experiments clearly demonstrate these advantages: 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.

6. REFERENCES

[1] B. Kovesi, D. Massaloux, and A. Sollaud, "A scalable speech and audio coding scheme with continuous bitrate flexibility," in *Proc. Int. Conf. on Acoustics, Speech, and Signal Processing*, May 2004, pp. I–273–I–276.

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

[3] 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.

[4] Z. Lu and W. A. Pearlman, "High quality scalable stereo audio coding," *http://www.cipr.rpi.edu/ pearlman/papers/*, 1999.

[5] 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.

[6] 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.

[7] 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*, Hong Kong, Apr. 2003, pp. V624–V627.

[8] J. Li, "Embedded audio coding (EAC) with implicit auditory masking," in *Proc. ACM on Multimedia*, Nice, France, Dec. 2002, pp. 592–601.

[9] S. A. Ramprashad, "High quality embedded wideband speech coding using an inherently layered coding paradigm," in *Proc. Int. Conf. on Acoustics, Speech, and Signal Processing*, Istanbul, Turkey, June 2000, pp. 1145–1148.

[10] Moving Picture Experts Group, "MPEG-4 Audio Version 2 (Final Committee Draft 14496-3 AMD1)," *ISO/IEC/JTC1/SC29/WG11 N2803*, 1999.

[11] 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.

[12] 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.

[13] "*http://www.iso.ch/iso/en/ittf/PubliclyAvailableStandards/ISO_IEC_14496-5_2001_Software_Reference/*".

[14] "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, *1997.*