A Dynamic Fine-Grain Scalable Compression Scheme With Application to Progressive Audio Coding

Stefan Strahl, *Member, IEEE*, Heiko Hansen, and Alfred Mertins, *Senior Member, IEEE*

***Abstract*—This paper studies the fine-grain scalable compression problem with emphasis on 1-D signals such as audio signals. Like in the successful 2-D still image compression techniques embedded zerotree wavelet coder (EZW) and set partitioning in hierarchical trees (SPIHT), the desired fine-granular scalability and high coding efficiency are benefited from a tree-based significance mapping technique. A significance tree serves to quickly locate and efficiently encode the important coefficients in the transform domain. The aim of this paper is to find such suitable significance trees for compressing dynamically variant 1-D signals. The proposed solution is a novel dynamic significance tree (DST) where, unlike in existing solutions with a single type of tree, a significance tree is chosen dynamically out of a set of trees by taking into account the actual coefficients distribution. We show how a set of possible DSTs can be derived that is optimized for a given (training) dataset. The method outperforms the existing scheme for lossy audio compression based on a single-type tree (SPIHT) and the scalable audio coding schemes MPEG-4 BSAC and MPEG-4 SLS. For bitrates less than 32 kbps, it results in an improved perceived audio quality compared to the fixed-bitrate MPEG-2/4 AAC audio coding scheme while providing progressive transmission and finer scalability.**

***Index Terms*—Audio coding, embedded coding, progressive compression, significance tree.**

I. Introduction

THE main attractive feature of a scalable compression algorithm is the possibility of progressive transmission. Scalability enables the receiver to decode the received signal at different fidelity levels, depending on the actual needs and available device capabilities, and it also allows one to adapt the data rate to the actual channel capacity. Certainly, such a feature is extremely desirable in packet-based networks, and it has been exploited to handle problems such as data-rate fluctuation, channel congestion or limited storage space.

For audio compression, the majority of classic encoders optimize on a single (although arbitrary) target compression ratio, striving to deliver the best quality given the bitstream length, or to deliver the shortest bitstream length given a constraint on quality. An example is the well-known MPEG2/MPEG4 advanced audio coder (MPEG-2/4 AAC) [1], which is a state-of-the-art audio compression tool that provides excellent quality at bitrates of 64 kbps per channel and also yields excellent performance, relative to the alternatives, at bitrates reaching as low as 16 kbps. Clearly, no scalability is offered from those conventional coders.

To cater for the scalability desire, a few scalable encoders that are organized in layers have been proposed and standardized. Namely, ITU-T G.727 [2] (5-, 4-, 3- and 2-bit/sample) for telephone bandwidth or ITU-T G.722 [3] (48/56/64 kbps) for wideband speech. More recently, MPEG-4 CELP [4] (2 kbps in the narrowband version and 4 kbps in wideband) and MPEG-4 BSAC [4] (with up to 1-kbps fine bitrate graduation) have been introduced. BSAC stands for bit slice arithmetic coding. In the MPEG-4 BSAC coder, a core layer produces the lowest bitrate and provides the minimum information to obtain a basic quality for the decoded signal, and several enhancement layers contain additional information that allows the decoder to improve the quality. The scalability is obtained by transmitting the core layer bitstream, combined with one or more enhancement layers. Clearly, the obtained granularity relies on the predefined bitrates allocated to the layers and the number of layers sent to the decoder. Typically, for achieving scalable bitrates between 16 and 64 kbps, up to 48 enhancement layers are used. The newest scalable audio coding scheme addition to the MPEG-4 standard is MPEG-4 SLS [5] which provides backward compatibility to MPEG-2/4 AAC while achieving a granularity of up to 0.4 kbps. SLS stands for scalable to lossless audio coding and the MPEG-4 SLS coder achieves scalability by using, similar to the MPEG-4 BSAC coding standard, a layered concept. The bitstream consists of an AAC core layer defining the lowest possible bitrate and a lossless enhanced (LLE) layer that produced the fine grain scalable to lossless portion of the lossless SLS bitstream. The LLE layer encodes the residual signal using a bitplane coding approach whose quantizer noise reproduces the perceptually optimized spectral shape of the AAC core layer's quantizer noise [6]. The MPEG-4 SLS standard also defines a computational less complex non-core mode for applications that require only lossless quality.

In all existing standardized scalable encoders, the scalability is obtained at the price of degradation in terms of performance when compared to fixed-rate schemes, and, in general, the finer the granularity is, the higher the loss is [7]. A few other,

Manuscript received January 12, 2009; revised September 24, 2009; accepted January 04, 2010. Date of publication February 05, 2010; date of current version October 01, 2010. This work was supported in part by the SFB/TRR 31 and in part by the International Graduate School for Neurosensory Science and Systems, Carl von Ossietzky University, Oldenburg. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Patrick A. Naylor.

S. Strahl and H. Hansen are with the Carl von Ossietzky University, D-26111 Oldenburg, Germany (e-mail: stefan.strahl@uni-oldenburg.de).

A. Mertins is with the Institute for Signal Processing, University of Lübeck, Ratzeburger Allee 160, D-23538 Lübeck, Germany.

Digital Object Identifier 10.1109/TASL.2010.2042129

nonstandard coders with fine bitrate scalability have been proposed as well. These are, for example, coders based on the SPIHT principle [8]–[13], which will be discussed in more detail below, the embedded wideband speech coder in [14], which is based on layered multimode transform predictive coding, the layered coder in [7], which orders and transmits parameters in terms of their significance, and the embedded audio coder in [15], which forms embedded streams for time segments of about 0.74-s duration. For the MPEG-4 SLS scheme, a prioritized bitplane coding has been proposed [16], that similar to the approach presented in this work, divides the frequency spectrum of the LLE layer into several regions and assigns these regions with coding priorities according to their respective energy levels.

Several of the previously mentioned scalable coders use bitplane coding (also called bit-slice coding), where the coefficients are transmitted layer by layer, starting with the layer of most significant bits. In the first encoding round, this provides coarse representations of the largest coefficients (largest in magnitude), and subsequent layers provide more accurate representations of the coefficients. For a larger set of coefficients to be encoded, in order to achieve efficient bitplane coding, it is advantageous to describe the bitplanes via position and value information, instead of transmitting value information alone in a straightforward manner. This is in particular interesting for sparse data where most of the coefficients are zero. One successful solution of this is the tree-based significance mapping technique [17], [18]. In these methods, for a set of coefficients to be encoded, by assuming a known coefficient significance/magnitude distribution in the form of trees, the coefficient position information is mapped into node-location information in the tree domain. Moreover, different significance-tree structures result in different compression efficiencies. Details will be given in Section II.

For 2-D coefficient-set compression, applying tree-based techniques has produced impressive advances in wavelet-based image compression. Its development could be traced back to the work of Lewis and Knowles [19], who used tree structures to exploit the statistical properties found in the pyramidal decomposition of natural images. The technique was further developed by Shapiro [17], who proposed an efficient method to combine the two techniques, bitplane coding and tree-based significant-coefficient selection (sorting), and applied them to the wavelet transform coefficients. This combination, called embedded zerotree wavelet (EZW) algorithm, was later refined by Said and Pearlman [18]. The according new algorithm, called set partitioning in hierarchical trees (SPIHT), is one of the state-of-the-art progressive image compression algorithms. It has been realized that the reason for the success of the SPIHT fine bitstream scalability, state-of-the-art compression performance, and reasonable computational complexity, is mainly attributed to the effective description of the significance map of wavelet coefficients. This has been confirmed from both empirical observations and theoretical analysis: in the experiment illustrated in [20], the SPIHT algorithm successfully captures not only the inter-band correlation but also the intra-band correlation. Theoretical support can be found in [21] and [22], where the statistical models between the magnitudes of wavelet coefficients in different scales and orientations were proved to be existent.

Inspired by the success in image compression, SPIHT-related coding techniques have been proposed for audio compression as well [8]–[13]. In all of these approaches, the tree structures have been fixed independent of the signals to be encoded and are based on the assumption that low-frequency components contain more energy than high-frequency ones. This assumption, however, does not hold for all frames of real-world audio signals, so that fixed trees can only be suboptimal.

To address this problem, in an earlier work, the authors of this paper proposed an adaptive tree-based significance mapping technique that used a fixed set of significance trees from which the optimal tree for each frame was selected [23], [24]. In the present paper, we present a novel, scalable compression scheme with a dynamic set of data-dependent significance trees, called dynamic significance tree quantization (DSTQ). As the compression performance of a tree depends on how well it matches the signal structure, the set of significance trees should be optimized for the applied signal class. We propose to derive such a set of data-dependent significance trees directly from the coefficient's distribution. An initial set of significance trees can be either learned for a general signal class (e.g., speech), or they can be optimized for the specific signal to be encoded. A dynamic adaptation of the set of trees is possible online without sending further side information if the encoder and decoder stage use the same learning algorithm. The proposed DSTQ algorithm can provide bitrate scalability at a granularity of one bit per frame, and has better performance than the existing SPIHT-related algorithms.

Another concept of scalable audio coding is the optimization at the encoding stage for a wide range of bitrates and types of input signals for an *a priori* known bitrate. For this problem, hybrid sound coding schemes have been proposed [25], using in parallel sinusoidal, transform or CELP coding modules and optimizing the respective bitrates or time segmentation using for example operational rate-distortion optimization. These scalable audio coders result in a bitstream with a fixed bitrate and are therefore different from the scalable audio coders like MPEG-4 BSAC, MPEG-4 SLS or our proposed coding schemes, where the bitrate can be changed after the encoding process, scaling only the bitstream itself.

This paper is organized as follows. To facilitate the later description of our algorithm, a brief overview of existing SPIHT-style algorithms (both in image and audio compression) is presented in Section II. Then, Section III describes our proposed dynamic significance tree method and develops the scalable compression scheme DSTQ. We present a data-driven approach to generate a set of optimal significance trees. To illustrate the compression performance, we compare in Section IV the achieved signal reconstruction performance with the SPIHT scheme for lossy audio compression based on a single-type tree and our previously proposed scheme using a fixed set of significance trees (CSTQ). We further evaluate the achieved perceptual quality for compression of quantizer indices compared with the fixed-bitrate MPEG-2/4 AAC and scalable MPEG-4 BSAC and MPEG-4 SLS audio coding schemes. Conclusions are given in Section V.

Notation

Matrices and vectors are printed in boldface, sets are printed in script alphabet and trees in fraktur alphabet. $\oplus$ denotes the addition of sets and $\mathbb{N}^+$ is the set of all positive integers excluding zero. $\lfloor x \rfloor$ denotes the greatest positive integer less than or equal to a given positive real number x. $[a,b] := \{x \in \mathbb{N}^+ | a \leq x \leq b\}$ represents the set of all positive integers between and including a and b.

II. Tree-Based Significance Mapping in SPIHT

A. SPIHT Algorithm in Image Compression

In this section, we give a brief summary of some characteristics of the SPIHT algorithm, introduced by Said and Pearlman in [18] for 2-D wavelet-based image compression.

Assume an original image is wavelet transformed to a 2-D coefficient array $\mathbf{X}$. Each element $X_{(i,j)}$ of $\mathbf{X}$ is called transform coefficient at coordinate (i,j) and represented in its binary form. Note that for efficient information transmission, the most significant value information should be transmitted first. To achieve this, the idea is to encode the coefficient value information in decreasing bitplane order. In particular, the sign and value of a coefficient are encoded only when the coefficient has become significant, that is when its most significant nonzero bit is located at the current or one of the previous bitplanes. This idea leads to another issue, which is the question of how to efficiently encode the coordinates of significant coefficients. A good solution is provided by the tree-based significance mapping technique.

A tree is a set of linked nodes that realizes a hierarchical data structure. Each node has at most one parent node and a set of zero or more children nodes. A node with zero child nodes is also called a leaf node and the root node is defined as the topmost node that has no parent node. The order of the tree defines the number of children of a node. A significance tree is generated by ordering all coefficients in the form of trees with the assumption that the coefficients closer to the roots of the trees will usually be more significant (i.e., larger in magnitude) than those at the leaves. In SPIHT coding of the wavelet coefficients of an image, such a tree is recursively generated by the parent-offspring relationship $\mathcal{O}(i,j) = \{(2i,2j),(2i,2j+1),(2i+1,2j),(2i+1,2j+1)\}$, where (i,j) is the coordinate of a parent and $\mathcal{O}(i,j)$ the set of coordinates of its offspring (also called direct descendants of the parent at node (i,j)). Each of the members of the set $\mathcal{O}(i,j)$ then has its own offspring, which are called indirect descendants with regard to the parent node (i,j). For the description of the algorithm, all the indirect descendants of the parent with coordinates (i,j) are gathered in the set $\mathcal{L}(i,j)$. Finally, a complete descendants set $\mathcal{D}(i,j)$ is defined as the sum of the direct and indirect descendants $\mathcal{D}(i,j) = \mathcal{O}(i,j) \oplus \mathcal{L}(i,j)$.

The above-mentioned relationship between parents and their offspring provides a natural link between wavelet coefficients in different frequency bands that belong to the same spatial location in an image. It holds for all coefficients, except the ones in the highest and lowest frequency bands, because the coefficients in the highest bands do not have any offspring, and in the lowest band, only three out of four coefficients have offspring. Fig. 1(a) gives an illustration.

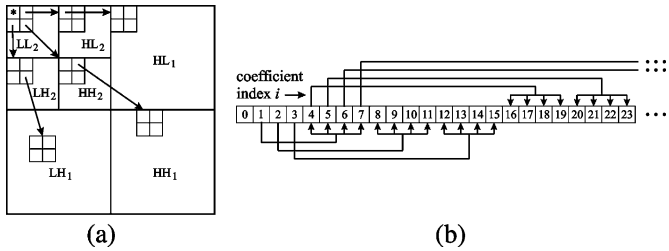

Fig. 1. Parent-offspring dependencies in SPIHT. (a) 2-D tree for a seven-band wavelet transform. The larger squares represent the frequency bands $(LL_2, HL_2, LH_2, HH_2, HL_1, LH_1, HH_1)$ of the transformed image, and the smaller ones represent the individual coefficients within the bands. The arrows show the links between parents and their offspring. (b) 1-D tree following the offspring rule $\mathcal{O}(i) = iN + [0, N-1]$ with $N = 4$.

Based on the significance tree, which is also called spatial orientation tree, the entire SPIHT algorithm performs iterative sorting and refinement passes to progressively encode the coefficient array $\mathbf{X}$. From the top bitplane $n_{\max} \in \mathbb{N}^+$, naturally decided by $2^{n_{\max}} > \max_{(i,j)}(|X_{(i,j)}|/2)$, each *sorting pass* is used to perform three actions: The first is to find all coefficients that are significant with respect to the current bitplane (these are coefficients that are larger in magnitude than the current threshold 2^n with $n = n_{\max}, n_{\max}-1, \ldots$). The second action is to transmit the significance-testing results and signs of those coefficients that have become significant against the current threshold. The final action is to update the initial spatial orientation trees by removing all significant coefficients and storing them separately. Here, the significance tests are performed on the basis of sets. If a set has become significant (at least one coefficient inside the set whose most significant bit locates at the current bitplane), a partitioning rule is used to partition the set into new subsets, then significance tests are performed on the new, smaller sets. This process continues until the significance test has been done for all significant sets, and the coordinates of all significant coefficients for the current bitplane have been identified. A succeeding *refinement pass* is used to transmit the current bitplane values for coefficients that are known to be significant from the previous bitplanes. The whole sorting and refinement-pass sequence is repeated until the desired bitrate is achieved or, in the case of lossless compression of finite-alphabet data, until all bitplanes have been transmitted.

B. SPIHT-Style Algorithm in Audio Compression

The idea of applying SPIHT-type significance trees to audio compression has been independently proposed in [9] and [8]. Both focused on compressing MDCT (modified discrete cosine transform) transformed audio signals and applied a parent-offspring relationship for the coefficient coordinates of the form $\mathcal{O}(i) = iN + [0, N-1]$ where $i \in \mathbb{N}^+$ is the parent coordinate and $N \in \mathbb{N}^+$ is the number of offspring. This tree choice is somehow related to the fact that most instruments produce harmonics of a fundamental frequency, so that correlations might exist between the coefficients and their harmonics. Because this type of significance tree was inspired by the SPIHT algorithm, we will refer to it as the SPIHT-style significance tree in the

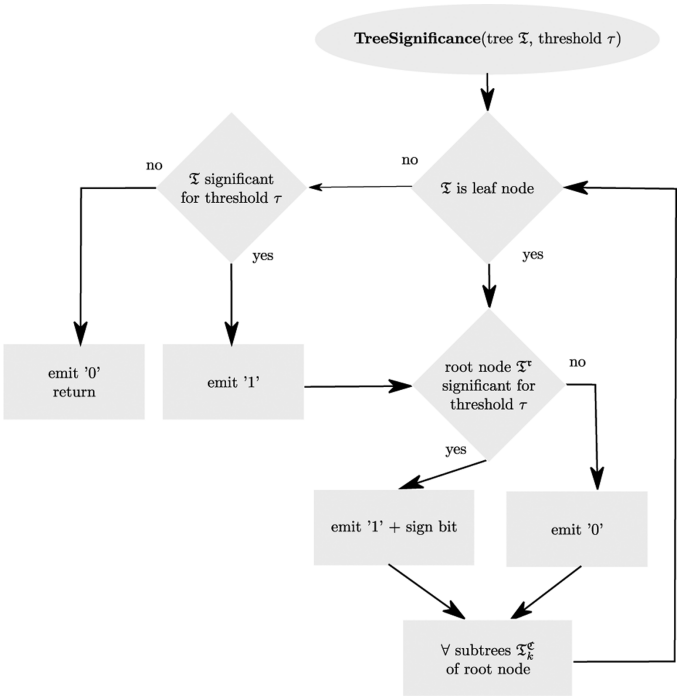

Fig. 2. Flowchart representation of Algorithm 1.

following. Fig. 1(b) illustrates the SPIHT trees for $N = 4$. In addition to SPIHT-related compression, additional perceptual significance tests were introduced in [11]. In this method, coefficients that were significant with respect to their magnitudes were only transmitted if they were also significant with respect to a masking threshold.

III. Description of the Dynamic Scalable Compression Scheme DSTQ

In this section, we consider the problem of constructing optimal significance trees for a 1-D transform vector and how to use them for compression. First we will explain the proposed DSTQ algorithm. Then we will present an approach to generate a data-dependent set of optimal significance trees.

A. DSTQ Algorithm

Let the vector $\mathbf{X} = (X_1, X_2, \ldots, X_M), M \in \mathbb{N}^+$ be the 1-D coefficient vector to be encoded, with the corresponding set of coefficient coordinates $\mathcal{M} = [1, M]$. Here the data in the coefficient vector $\mathbf{X}$ is not specified. It can, for example, be real-valued signal samples, transform coefficients or integer-valued quantizer indices in a frame of audio. Similar to the SPIHT algorithm, our DSTQ algorithm encodes the coefficients subsequently bitplane for bitplane, commonly starting from their most significant and continuing to their least significant bitplane. The most significant bitplane n_{MSB} is determined by $2^{n_{\mathrm{MSB}}} > \max_{i \in \mathcal{M}}(|X_i|/2)$. DSTQ also distinguishes between a *sorting pass* (to select significant coefficients by tree-based significance mapping and output "position" bits) and a *refinement pass* (to output "value" bits).

Now let us assume that the coefficient-position information $\mathcal{M}$ is mapped to a significance tree $\mathfrak{T}$. Then the so-called *sorting pass* performs the following significance-tree tests for the current processed bitplane n

$$S(n) = \begin{cases} 1, & \text{if } \max_{i \in \mathcal{M}}(|X_i|) \geq 2^n \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

The significance-tree test $S(n)$ can be performed[1] calling the following pseudocode *TreeSignificance* $(\mathfrak{T}, 2^n)$:

Algorithm 1 TreeSignificance(tree $\mathfrak{T}$, threshold τ)

1: **if** $\mathfrak{T}$ is not a leaf node **then**
2: **if** $\mathfrak{T}$ is insignificant with respect to τ**then**
3: emit "0" and return
4: **else**
5: emit "1"
6: **end if**
7: **end if**
8: $\mathfrak{T}^{\mathfrak{r}} \Leftarrow$ root node of $\mathfrak{T}$
9: **if** $\mathfrak{T}^{\mathfrak{r}}$ is significant with respect to τ **then**
10: emit "1" and sign bit
11: **else**
12: emit "0"
13: **end if**
14: $\mathfrak{T}_k^{\mathfrak{C}} \Leftarrow k$th child subtree of $\mathfrak{T}^{\mathfrak{r}}$
15: **for all** $\mathfrak{T}_k^{\mathfrak{C}}$ **do**
16: Call **TreeSignificance**($\mathfrak{T}_k^{\mathfrak{C}}$, τ)
17: **end for**.

See also Fig. 2 for a flow-chart representation of Algorithm 1. The important feature of this kind of coefficients-position mapping technique can be seen in the pseudocode of Algorithm 1 in the lines 2 and 3. We can save bit costs whenever we encode an insignificant tree (i.e., a tree with $S(n) = 0$) with only one bit "0" instead of coding all the insignificant coefficients of the tree one-by-one. Even though there is additional cost for transmitting the significance-tree test result for significant trees, due to the savings that occur when a tree is insignificant, the method is, in general, more efficient than the straightforward one-by-one coding.

The proposed DSTQ algorithm compresses the coefficient set $\mathbf{X}$ in the order of threshold-by-threshold. At each bitplane, in the sorting pass, the procedure *TreeSignificance* is applied based on the dynamically selected local significance tree $\mathfrak{T}$. The coefficient positions that become significant in the current bitplane are determined and moved into a respective list of significant coefficients (LSC). In the refinement pass, also sequentially, we output the current bitplane values of those coefficients that became significant in the previous bitplanes. Then, we move to the next lower bitplane, and the sequence of sorting and refinement passes is repeated. In this way, the DSTQ algorithm achieves encodings with finer and finer quantization steps by progressively transmitting the binary representation of the coefficients

[1]The partitioning rule is slightly simplified compared to the 2-D one in SPIHT [18]

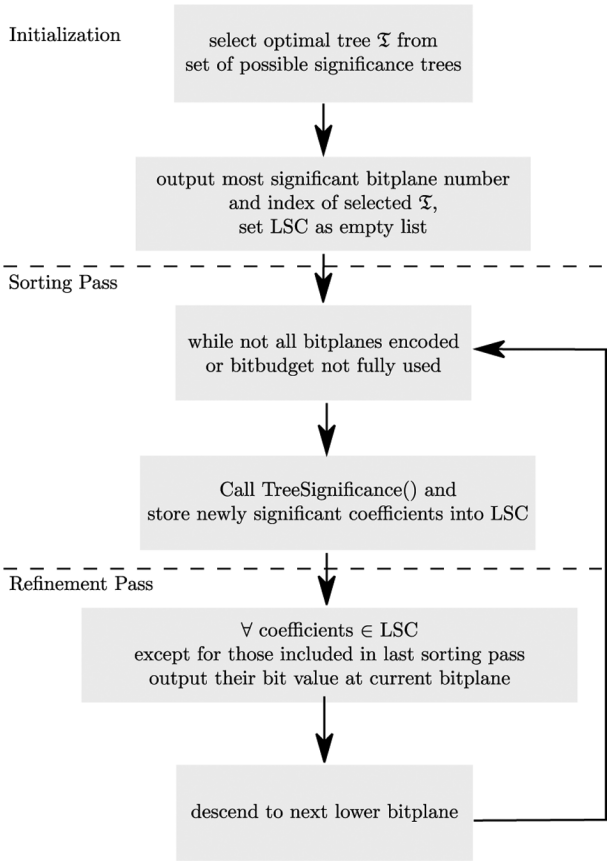

Fig. 3. Flowchart representation of Algorithm 2.

and yields a bit-wise scalable compression technique. The entire DSTQ algorithm is described as Algorithm 2 (see Fig. 3 for a flow-chart representation).

Algorithm 2 DSTQ(coefficients $\mathbf{X}$, coordinates $\mathcal{M}$)

1: *Dynamic Tree Generation:*
2: select optimal tree $\mathfrak{T}$ from set of possible significance trees
3: *Initialization:*
4: output $n = \lfloor \log_2(\max_{i\in\mathcal{M}}\{|X_i|\})\rfloor$;
5: output index of selected $\mathfrak{T}$
6: set LSC as an empty list.
7: **while** $n \geq 0$ and bit budget is not fully utilized **do**
8: *Sorting Pass:*
9: call *TreeSignificance* * $(\mathfrak{T}, 2^n)$
10: store all newly significant coefficients into the LSC
11: *Refinement Pass:*
12: **for all** coefficient in LSCs except those included in the last sorting pass **do**
13: output the nth bit
14: **end for**
15: $n = n - 1$
16: **end while**.

The previous definition of *TreeSignificance* described a simplified version for comprehensibleness. In the *DSTQ* algorithm, a modified version *TreeSignificance* * is used that only emits test results for nodes that have not become significant in a previous bitplane. These coefficients are encoded in the refinement pass and their removal from the set of tested coefficient results in larger insignificant trees. The process is repeated until the desired rate (bit budget for compressing the coefficient set) is achieved, or, in case of lossless compression, all bits in all coefficients have been encoded. Like with the technique used in [18], to obtain the desired decoder's algorithm that duplicates the encoder's execution path, we simply have to replace the words "output" by "input" in the pseudo code.

The amount of side information required to transmit which significance tree has been selected is relatively low compared with the number of saved bits due the optimized significance tree. This will be confirmed by experimental results in Section IV, where we compare the performance of single fixed trees with the one of dynamically selected trees.

B. Data-Driven Generation of Significance Trees

As stated before, in wavelet-based still image coding, due to the properties of natural images, the significance information can be well captured within a single type of significance tree. For audio, however, where the frequency content can change dramatically over time, there exists no single tree that captures the significance information of all frames equally well. We therefore propose to dynamically select the best significance tree for each frame from a given set of possible trees. To efficiently construct such a set of significance trees, we suggest to directly use the information available from the coefficients distribution. We will first recapitulate some theoretical results on optimal significance trees for a given coefficients distribution.

For a significance tree $\mathfrak{T}$ the root node is denoted as $\mathfrak{T}^{\mathfrak{r}}$ and the kth child subtree of $\mathfrak{T}^{\mathfrak{r}}$ is represented by $\mathfrak{T}_k^{\mathfrak{C}}$. The probability of significance of a tree with respect to the bitplane n is denoted as $P(\mathfrak{T}, n)$. The probability of significance of the root node at a given bitplane is written as $p(\mathfrak{T}^{\mathfrak{r}}, n)$. Then the average significance mapping cost of a tree $\mathfrak{T}$ for a bitplane n is given by

$$C(\mathfrak{T}, n) := 1 + P(\mathfrak{T}, n)\left(1 + \sum_k C\left(\mathfrak{T}_k^{\mathfrak{C}}, n\right)\right) \qquad (2)$$

with

$$P(\mathfrak{T}, n) = 1 - (1 - p(\mathfrak{T}^{\mathfrak{r}}, n)) \prod_k \left(1 - P\left(\mathfrak{T}_k^{\mathfrak{C}}, n\right)\right). \qquad (3)$$

From (2), it follows that the encoding cost for a significance tree depends recursively on the probability of significance (3) of its child subtrees and the number of these recursive steps executed. This leads to the important conclusion that the optimal sequence of coefficient positions is in descending order of their magnitude, resulting in the largest possible insignificant subtrees for each bitplane.

We will illustrate this using the toy example given in Table I. The sorting tree for this example, shown in Fig. 4,

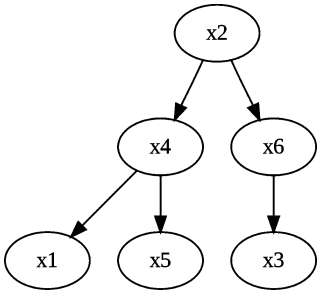

Fig. 4. Significance tree of the 1-D example which is its sorting tree.

TABLE I
BINARY REPRESENTATION OF THE 1-D EXAMPLE SIGNAL

Coefficient		Bitplane		
Position	Value	3rd	2nd	1th
X_1	2	0	1	0
X_2	4	1	0	0
X_3	0	0	0	0
X_4	3	0	1	1
X_5	1	0	0	1
X_6	0	0	0	0

is mapping the coefficient positions depth-first in the order $\mathcal{M} = \{X_2, X_4, X_1, X_5, X_6, X_3\}$. An optimal tree would transmit for a bitplane all significant coefficients first, followed by the remaining insignificant coefficients. For the example, its sorting tree maps in the third bitplane the significant coefficient X_2 first and *TreeSignificance* will emit $\{1\ 1\ s\}$ to the bitstream with s being the sign bit. The remaining coefficients of the bitplane are fully described by the test results for the subtrees with the root nodes X_4 and X_6. The bitstream encoding the MSB of the example is therefore $\{1\ 1\ s\ 0\ 0\}$. As for every bitplane its significant coefficient positions are moved to a list of significant coefficients (LSC) they are excluded from the sorting pass in the following bitplanes. Thereby all significant coefficients in the next bitplane will be of lower magnitude than the coefficients that became significant in the current bitplane. The sorting tree will map these coefficient first, in the example X_2 is skipped in the next sorting pass and the subtree with the root node X_4 is tested first, adding $\{1\ 1\ s\}$ to the bitstream. Then, the leaves mapping X_1 and X_5 are encoded and the insignificant coefficients X_6 and X_3 are encoded with a single 0 bit. In the refinement pass the next bit representing X_2 is transmitted, resulting in the overall bitstream $\{11s0011s1s000\}$. In the last bitplane, the sorting pass emits the test result for the subtree with the root node X_6. With the final refinement pass the bitstream of the example is $\{1\,1\,s\,0\,0\,1\,1\,s\,1\,s\,0\,0\,0\,0\,0\,1\,0\,1\}$.

It can be concluded that an optimal significance tree for a given 1-D coefficient vector can be derived by computing its sorting tree. In audio coding applications, the signal is commonly divided into frames of fixed length. To derive a set of optimal significance trees for such frames, we define a training dataset, and the sorting trees for every frame of the training dataset are computed. If the DSTQ coding scheme is applied for an entire signal class, for example for human speech in a telecommunication scenario, the training dataset would be a speech database. If the signal to be encoded is known *a priori*, like it is the case for storing digital audio, the signal itself can be used as the training dataset. We propose the following simple algorithm to learn a set of significance trees that is optimized for the training dataset. First, we encode the F frames of the training dataset with all available F sorting trees derived from these frames. The performance of every tree is measured using the lengths of the resulting bitstreams needed to achieve lossless encoding. These derived metric values are stored into a $F \times F$ matrix $\mathbf{M}$. It is to note that the DSTQ algorithm has a low complexity—a simple C prototype implementation on a standard workstation was able to test 10 000 trees per second for a frame length of 1024 samples. We then use the following algorithm to reduce the number of significance trees to K trees.

Algorithm 3 Derive set of K optimal significance trees

1: find in $\mathbf{M}$ for every frame the tree achieving the best performance and store their ID in $\mathbf{T} \in \mathbb{N}^F$
2: **while** (number of unique trees in $\mathbf{T}$) $> K$ **do**
3: get for every tree in $\mathbf{T}$ its next best tree for the according frame from $\mathbf{M}$ and store its ID in $\mathbf{T}' \in \mathbb{N}^F$
4: replace in $\mathbf{T}$ the tree whose next best tree in $\mathbf{T}'$ is already used in $\mathbf{T}$ and which results in the smallest performance degeneration for its frame.
5: if no such tree is found in $\mathbf{T}'$ search for the second next best/third next best/…tree.
6: **end while**.

Note that instead of the length of the bitstream achieving lossless compression a perceptual metric could be used, and the computational complexity could be reduced by limiting the iterations of the DSTQ scheme to the first n bitplanes or to a given bitrate. If both the encoder and the decoder know the F previously decoded frames to a certain minimal reconstruction level, it is possible to use this information to derive new optimal significance trees after each audio frame which can replace trees in the set that have seldom been selected for the past frames. Another alternative to dynamically update the set of significance trees is to use a side-information channel that continuously transmits updates to the set of significance trees.

IV. EXPERIMENTAL RESULTS

We applied our proposed DSTQ scheme to the compression of audio signals, and in this section, we compare our method with the existing algorithms SPIHT [8], [9] and CSTQ [23] for lossy audio compression. In addition, we combined our method with the state-of-the-art MPEG-2/4 AAC compression scheme by replacing the Huffman coding stage for quantizer-index compression by the SPIHT, CSTQ, and DSTQ algorithms. Performance comparisons for this AAC-related scheme are made with the standardized MPEG-2/4 AAC scheme with low-complexity profile (fixed bitrate) and the scalable MPEG-4 BSAC and MPEG-4 SLS schemes.

A. Comparison With Schemes Using a Priori Fixed Trees

In this experiment, we compared the compression performance of our proposed DSTQ algorithm with the single-tree SPIHT algorithm and the CSTQ coding scheme which uses a

TABLE II
SEGMENTAL SNRS IN dB FOR THE SVEGA TEST SIGNAL ENCODED AT BITRATES BETWEEN 16–96 kbps USING DIFFERENT ALGORITHMS

Algorithm	tree ID coding cost	bitrate					
		16kbps	32kbps	48kbps	64kbps	80kbps	96kbps
SPIHT (1 tree)	0 bit	15.121	19.268	22.817	26.073	29.081	31.945
CSTQ (65536 trees)	16 bit	16.116	20.641	24.352	27.710	30.846	33.873
DSTQ (512 trees)	9 bit	17.865	22.577	26.329	29.638	32.627	35.348
DSTQ (256 trees)	8 bit	16.536	20.607	23.939	26.963	29.787	32.419
DSTQ (128 trees)	7 bit	15.786	19.602	22.771	25.680	28.427	31.018
DSTQ (64 trees)	6 bit	15.345	19.025	22.132	24.990	27.698	30.266
DSTQ (32 trees)	5 bit	15.073	18.687	21.759	24.602	27.295	29.865

set of *a priori* fixed trees [23]. The CSTQ algorithm separates an audio frame into eight segments and selects for each segment a tree that assumes a descending, ascending, convex or concave coefficient-magnitude behavior. This results in a set of 65 536 possible trees for every frame. The *a capella* song "Tom's Diner" by Suzanne Vega was transformed with the MDCT filterbank with $M = 1024$ frequency bands and then encoded with the SPIHT, CSTQ, and DSTQ algorithms, respectively. All coding schemes were performed with significance trees having a tree order of four. For the DSTQ algorithm, a full search over all trees was performed for every audio frame, selecting the tree that results in the shortest bitstream for a lossless encoding. Sets of optimal significance trees consisting of 32, 64, 128, 256, and 512 trees have been derived as described in Algorithm 3. To allow for a direct comparison of the significance trees, the same tree-selection algorithm was used for CSTQ in contrast to [23], where a less complex tree selection algorithm was applied. The test bitrates were varied between 16 and 96 kbps. The according frame bit budget R_f was computed as $R_f = \lfloor R \cdot M/Fs \rfloor$, where R is the required bitrate in bits per second, and Fs is the sampling rate in Hz. As a quality measure for the compression performance, the segmental signal-to-noise ratio (segSNR) was used, which was computed as follows:

$$\text{segSNR} = \frac{1}{N}\sum_{j=0}^{N-1} 10\log_{10}\frac{\sum_{i=1}^{M}\left(X_i^{(j)}\right)^2}{\sum_{i=1}^{M}\left(X_i^{(j)}-\hat{X}_i^{(j)}\right)^2} \quad (4)$$

where $X_i^{(j)}$, $i \in [1, M]$ are the transform coefficients in frame j and $\hat{X}_i^{(j)}$ are the corresponding reconstructed coefficients. N is the number of frames. A psychoacoustic analysis was not carried out for this experiment, because all schemes applied the same MDCT and significance tests (i.e., the same thresholds), so that, in this special case, the segmental SNR indeed allows for a comparison between the different significance tree-related compression techniques.

The results presented in Table II show that algorithms using a set of possible significance trees achieve a better segmental SNR than single-tree algorithms like SPIHT. The number of saved bits due to the selection of significance-trees being optimal for the current frame is larger than the amount of side information needed to transmit which significance-tree has been selected. It further shows that using a data-driven approach to construct a set of trees being optimal for a specific class of signals (DSTQ), a higher compression performance can be achieved with a smaller set of possible significance trees compared to a model-driven approach (CSTQ). For bitrates equal to or less than 32 kbps, a set of 256 optimized significance trees achieved a similar performance as the set of 65 536 model-driven trees (CSTQ). For higher bitrates, a set of 512 learned significance trees showed a superior signal reconstruction performance than CSTQ. For digital audio storage applications, the derived optimal set can be placed at the beginning of the media and only the ID of the selected tree needs to be transmitted for every audio frame. Using a simple run-length encoding we needed at most 5000 bits to encode a significance tree for a frame length of 1024 samples. This would result in a maximal cost of 2.44 MB for 512 trees which is below 0.5% of the capacity of a standard audio CD. And for applications like speech coding it is not necessary to transmit the learned set of significance trees, as an *a priori* learned set of significance trees can be stored in the decoding device. Also a constant update of the significance tree set can be performed as proposed in Section III-B.

B. Comparison With MPEG-2/4 AAC, MPEG-4 BSAC, and MPEG-4 SLS

In this section, we compare the perceptual quality of our proposed DSTQ scheme with the single-tree SPIHT algorithm, the CSTQ coding scheme, the standardized MPEG-2/4 AAC fixed-bitrate encoder and the MPEG-4 BSAC and MPEG-4 SLS scalable encoders, respectively. Within the MPEG-2/4 AAC compression scheme, a flexible Huffman codebook selection (from 11 predesigned Huffman codebooks) is adopted to losslessly compress the quantizer indices. The MPEG-4 BSAC coder uses an alternative noiseless coding method (bit slice arithmetic coding instead of Huffman coding), with the rest of the processing (filterbank, psychoacoustic model, etc.) being identical to MPEG-2/4 AAC. The MPEG-4 BSAC coder is designed to support scalability with nearly transparent sound quality at 64 kbps and graceful degradation at lower bitrates and performs best in the range of 40 kbps to 64 kbps. The MPEG-4 SLS coder realizes a two-layer structure with a Huffman encoded AAC core and a lossless enhancement bitstream which is encoded using context-based bitplane arithmetic coding and a special entropy encoding mode for low-energy frames. MPEG-4 SLS is designed to achieve lossless coding at a bitrate of approximately 350 kbps/channel and nearly transparent sound quality at approximately 64kbps as a result of the AAC core layer. In our experiments, MPEG-2/4 AAC and MPEG-4 BSAC compression procedures were implemented based on the MPEG-2/4 reference software (2001 Edition) with available source codes on [26]. The MPEG-4 SLS coding scheme was implemented using the available source code on [27]. Due to the poor performance of the reference software we further included

a state-of-the-art MPEG-2/4 AAC implementation (Nero AAC codec 1.3.3.0, http://www.nero.com) in our test setup.

For a fair comparison, we adopted the basic MPEG-2/4 AAC encoding process before noiseless coding for our DSTQ coder as well. That is, we kept the MPEG-2/4 AAC scheme unchanged up to the point where Huffman coding is employed, but instead of using Huffman codebooks, our proposed DSTQ algorithm was employed for quantizer-index compression. In detail, the MDCT coefficients that have been quantized according to the psychoacoustic model are not Huffman encoded but compressed with the DSTQ coding scheme. The resulting bitstream consists of the MPEG-2/4 sideband information, e.g., the chosen scalefactors and the DSTQ coefficients that take up the same space in the new bitstream as the Huffman coefficients did in the original MPEG-2/4 AAC bitstream. This combined scheme will be referred to as the AAC-DSTQ scheme in the following. The CSTQ and SPIHT coding schemes have been realized in the same manner and will be referred to as AAC-CSTQ and AAC-SPIHT. To be able to compare the audio-coding performance of MPEG-4 SLS at low bit rates we chose for the AAC core layer a bitrate of 16 kbps. This allowed also a direct comparison with MPEG-4 BASC whose base layer is encoded at 16 kbps. It is to note that for MPEG-4 SLS higher AAC core layer bitrates have not been investigated in this work as the main focus is on coding scenarios with a scalability form low up to high bitrates which leads to the necessity of a low-bitrate AAC core in the MPEG-4 SLS scheme.

We evaluated the audio coding schemes using a set of five test files with different characteristics. From the SQAM test material [28] we selected a male German speaker (track #53), a recording of a harpsichord as an example for a strong harmonic sound structure (track #40), and a castanets recording (track #27) consisting of sharp attacks. We further selected the pop song "Mountains O'Things" by Tracy Chapman and the *a capella* song "Tom's Diner" by Suzanne Vega.

In the first experiment we used the MPEG-2/4 reference software. We applied AAC-DSTQ, AAC-CSTQ, and AAC-SPIHT to encode the quantizer indices derived by the reference software at 64 kbps, matching the maximum bitrate of MPEG-4 BSAC, and then truncated the bitstream to the different target bitrates. In the MPEG-4 SLS coding scheme the AAC core bitrate was set to 16 kbps to allow for a minimal bitrate of 16 kbps. We measured the perceived audio quality of the decoded audio signals relative to the original test signal using a model of auditory perception (PEMO-Q) [29]. The estimated perceived audio quality was mapped to a single quality indicator, the Objective Difference Grade (ODG) [30]. This is a continuous scale from 0 for "imperceptible impairment," -1 for "perceptible but not annoying impairment," -2 for "slightly annoying impairment," -3 for "annoying impairment" to -4 for "very annoying impairment." For the AAC-DSTQ, AAC-CSTQ, and AAC-SPIHT coders, the bitstreams in each frame consisted of the side information produced by the MPEG-2/4 AAC encoder when the total bit rate was selected to be 64 kbps, the tree-selection bits if applicable, and the embedded bitstream truncated to meet the target bitrates. It is to note that the encoding for AAC-DSTQ, AAC-CSTQ, and AAC-SPIHT was performed thereby only once for the highest bitrate and that the decoding at the lower bitrates was performed by simply truncating this bitstream to the desired bitrate. For the MPEG-2/4 AAC coder the complete encoding and decoding process was repeated for all lower bitrates. The tree order for all significance-tree based coding schemes was set to four. All signals were encoded as mono signals and temporal noise shaping was not used.

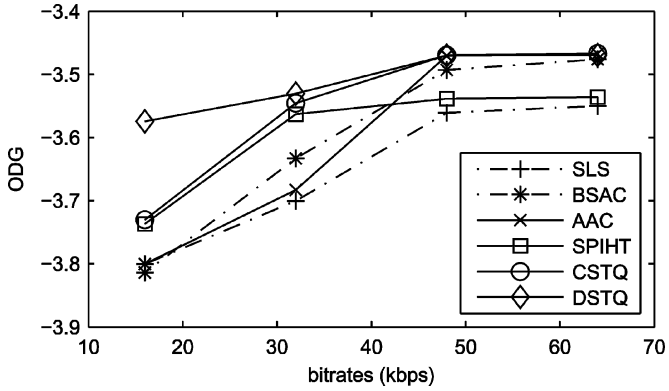

Fig. 5. MPEG-2/4 reference software. Objective difference grades for the set of audio test files encoded with MPEG-4 SLS, MPEG-4 BSAC, MPEG-2/4 AAC and AAC-SPIHT, AAC-CSTQ, and AAC-DSTQ using quantizer indices derived at 64kbps.

From the resulting objective difference grades for the set of audio test files shown in Fig. 5, we can see that MPEG-2/4 AAC, MPEG-4 BSAC, AAC-CSTQ, and AAC-DSTQ achieved a similar perceived audio quality from 64 to 48 kbps with MPEG-4 BSAC showing a marginally lower perceived audio quality at 48 kbps. For AAC-SPIHT lower ODGs were derived from 64 to 48 kbps that showed a parallel trend to the ODGs of AAC-CSTQ and AAC-DSTQ. MPEG-4 SLS resulted in the lowest ODGs from 64 to 32 kbps. MPEG-2/4 AAC resulted in the second lowest perceived audio quality for 32 kbps, followed by MPEG-4 BSAC, AAC-SPIHT, AAC-CSTQ, and AAC-DSTQ. For 16 kbps MPEG-4 BSAC resulted in the lowest ODG and a similar perceived audio quality was derived for MPEG-4 SLS and MPEG-2/4 AAC. AAC-DSTQ showed for all bitrates the best perceived audio quality. Interestingly the MPEG-2/4 AAC coder of the reference implementation achieved low ODGs for bitrates below 48 kbps despite the fact that it was allowed to optimize the quantizer indices for every target bitrate separately in contrast to the other coding schemes. It showed that the rate loop of the MPEG-2/4 AAC reference implementation did not optimally utilize the available bit budget at lower bitrates.

Therefore, we conducted a second experiment using the state-of-the-art Nero AAC encoder. We newly encoded the set of audio test files using MPEG-2/4 AAC Nero and analogous to the previous experiment these quantizer indices derived for 64 kbps were encoded using CSTQ and DSTQ. We further used a reduced number of audio coding schemes (MPEG-2/4 AAC, CSTQ-AAC, and DSTQ-AAC) to allow the evaluation of the perceived audio quality with the resource-demanding subjective listening test method MUSHRA (multi stimuli with hidden reference and anchor points) [31]. The tests were performed using AKG K240 headphones, Creative Sound Blaster Audigy sound cards and the ABC/Hidden Reference Audio Comparison Tool [32]. The 12 listeners performed a training

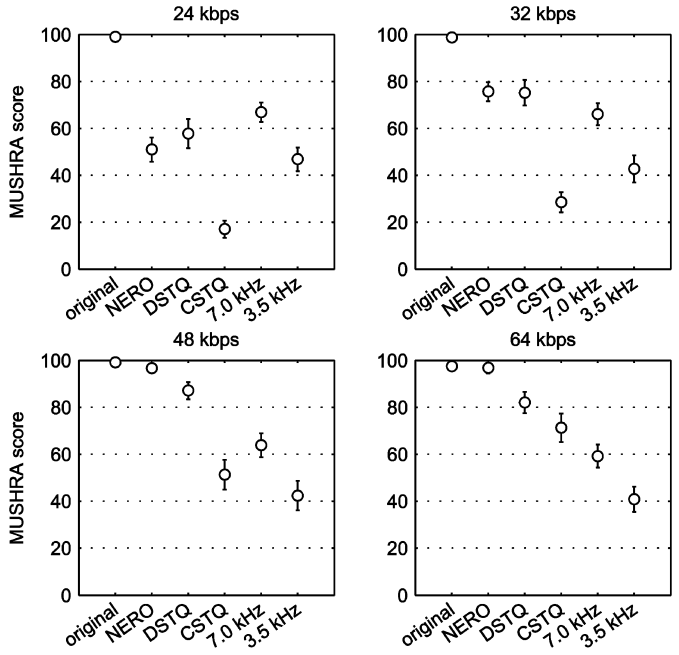

Fig. 6. MPEG-2/4 AAC Nero implementation. Results of MUSHRA listening tests for the set of audio test files encoded with MPEG-2/4 AAC Nero, AAC-DSTQ, and AAC-CSTQ using quantizer indices derived at 64 kbps and the hidden reference and anchors. Error bars denote 95% confidence intervals for mean.

session of approximately 15 min and had the opportunity to adjust the playback level only within this training period. Test instructions explained the user interface of the software and how to give the scores on the quality scale from 1 (bad) to 100 (excellent). The chosen anchor points were low-pass-filtered originals with cutoff frequencies of 3.5 and 7 kHz.

From the resulting MUSHRA scores for the set of audio test files shown in Fig. 6 we can see that for 64 kbps MPEG-2/4 AAC Nero achieved a transparent audio coding quality, followed by AAC-DSTQ, AAC-CSTQ and the reference implementation of MPEG-2/4 AAC. The same ranking order was derived for 48 kbps. For 32 kbps MPEG-2/4 AAC Nero and AAC-DSTQ achieved a similar MUSHRA score and for 24 kbps AAC-DSTQ achieved the best perceptual audio quality, followed by MPEG-2/4 AAC Nero and AAC-CSTQ and the reference implementation of MPEG-2/4 AAC achieved the lowest MUSHRA scores.

Overall, the perceptual quality for AAC-DSTQ, especially at low bitrates, was better than for the competing scalable coding schemes and for bitrates below 32 kbps, a higher perceptual coding quality than the non-scalable MPEG-2/4 AAC coding scheme was achieved. This indicates that the signal-dependent significance trees of the DSTQ scheme can reconstruct the important coefficients earlier than audio coding schemes assuming more general coefficient distributions.

V. Conclusion

The fine-grain scalable 1-D signal compression problem has been addressed in this paper. While in almost all existing SPIHT-related algorithms a single-type significance tree has been adopted for sorting significant coefficients and transmitting position information, we have proposed a novel dynamic significance tree technique, which learns optimal tree structures for 1-D signal compression. Based on the selection of an optimal tree from a learned set of significance trees, a compression scheme called DSTQ has been developed providing high compression quality and bitrate scalability. Further, we applied our proposed scheme to audio signal compression. Here, the advantages of our proposed scheme are clearly demonstrated: the method outperforms the existing SPIHT-like algorithm and yields competitive results for compressing quantizer indices in the MPEG-2/4 AAC audio compression scheme, yet with fine-grain scalability.

Acknowledgment

The authors would like to thank the anonymous reviewers for their constructive comments and corrections and the MUSHRA listening test participants for their effort. S. Strahl would like to thank A. Klinge for her support on this manuscript.

References

[1] K. Brandenburg, O. Kunz, and A. Sugiyama, "MPEG-4 natural audio coding," *Signal Process.: Image Commun.*, vol. 15, no. 2, pp. 423–444, 2000.

[2] "5-, 4-, 3- and 2-bit/sample embedded adaptive differential pulse code modulation (ADPCM)," ITU Publication, 1990, ITU-T Rec. G.727.

[3] "7 kHz audio coding within 64 kbit/s using sub-band adaptive differential pulse code modulation (SB-ADPCM)," ITU Publication, 1988, ITU-T Rec. G.722.

[4] *MPEG-4 Audio Version 2*, ISO/IEC 14496-3:1999/Amd.1, 1999, ISO/MPEG.

[5] *Scalable Lossless Coding (SLS)*, ISO/IEC 14496-3:2005/Amd.3:2006, 2006, ISO/MPEG.

[6] R. Yu, R. Geiger, S. Rahardja, J. Herre, X. Lin, and H. Huang, "MPEG-4 scalable to lossless audio coding," in *Proc. AES 117th Conv.*, San Francisco, CA, Oct. 2004, preprint 6183.

[7] B. Kovesi, D. Massaloux, and A. Sollaud, "A scalable speech and audio coding scheme with continuous bitrate flexibility," in *Proc. Int. Conf. Acoust., Speech, Signal Process. (ICASSP'04)*, Montreal, QC, Canada, May 2004, vol. 1, pp. I-273–6.

[8] C. Dunn, "Efficient audio coding with fine-grain scalability," in *Proc. AES 111th Conv.*, New York, Sep. 2001, preprint 5492.

[9] M. Raad, A. Mertins, and I. Burnett, "Audio coding based on the modulated lapped transform (MLT) and set partitioning in hierarchical trees," in *Proc. 6th World Multiconf. Syst., Cybern., Inform. (SCI 2002)*, Orlando, FL, Jul. 2002, vol. 3, pp. 303–306.

[10] M. Raad and A. Mertins, "From lossy to lossless audio coding using SPIHT," in *Proc. 5th Int. Conf. Digital Audio Effects (DAFx-02)*, Hamburg, Germany, Sep. 2002, pp. 245–250.

[11] M. Raad, A. Mertins, and I. S. Burnett, "Scalable to lossless audio compression based on perceptual set partitioning in hierarchical trees (PSPIHT)," in *Proc. 4th Int. Conf. Multimedia Expo (ICME 2003), Reprinted From ICASSP'03*, Baltimore, MD, Jul. 2003, vol. 3, pp. 393–396.

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

[13] Z. Lu and W. A. Pearlman, "High quality scalable stereo audio coding," 1999 [Online]. Available: http://www.cipr.rpi.edu/~pearlman/papers/scal_audio.ps.gz

[14] S. A. Ramprashad, "High quality embedded wideband speech coding using an inherently layered coding paradigm," in *Proc. Int. Conf. Acoust., Speech, Signal Process. (ICASSP'00)*, Istanbul, Turkey, Jun. 2000, pp. 1145–1148.

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

[16] Li Te, S. Rahardja, and S. N. Koh, "Frequency region-based prioritized bit-plane coding for scalable audio," *IEEE Trans. Audio, Speech, Lang. Process.*, vol. 16, no. 1, pp. 94–105, Jan. 2008.

[17] J. M. Shapiro, "Embedded image coding using zerotrees of wavelet coefficients," *IEEE Trans. Signal Processing*, vol. 41, no. 12, pp. 3445–3462, Dec. 1993.
[18] A. Said and W. A. Pearlman, "A new, fast and efficient image codec based on set partitioning in hierarchical trees," *IEEE Trans. Circuits Syst. for Video Technol.*, vol. 6, no. 3, pp. 243–250, Jun. 1996.
[19] A. S. Lewis and G. Knowles, "A 64 kb/s video codes using the 2-D wavelet transform," in *Proc. Data Compression Conf.*, Snowbird, UT, 1991, pp. 196–201.
[20] Z. Liu and L. J. Karam, "Quantifying the intra and inter subband correlations in the zerotree-based wavelet image coders," in *Conf. Rec. 36th Asilomar Conf. Signals, Syst., Comput.*, Pacific Grove, CA, Sep. 2002, pp. 1730–1734.
[21] E. P. Simoncelli, "Statistical models for images: Compression, restoration and synthesis," in *Conf. Rec. 35th Asilomar Conf. Signals, Syst., Comput.*, Pacific Grove, CA, Nov. 1997, pp. 673–678.
[22] X. Li and X. Zhuang, "The decay and correlation properties in wavelet transform," Tech. Rep., Univ. Missouri-Columbia, 1997.
[23] H. Zhou, A. Mertins, and S. Strahl, "An efficient, fine-grain scalable audio compression scheme," in *Proc. AES 118th Convention*, Barcelona, Spain, May 2005, preprint 6435.
[24] S. Strahl, H. Zhou, and A. Mertins, "An adaptive tree-based progressive audio compression scheme," in *IEEE Workshop Applicat. Signal Process. Audio Acoust. (WASPAA'05)*, New Paltz, NY, 2005, pp. 219–222.
[25] N. H. Schijndel *et al.*, "Adaptive RD optimized hybrid sound coding," *J. Audio Eng. Soc.*, vol. 56, no. 10, pp. 787–809, 2008.
[26] *ISO/IEC 14496-5:2001—Information Technology—Coding of Audio-Visual Objects—Part 5: Reference Software*, ISO/IEC 14496-5:2001, 2010, ISO/MPEG.
[27] *ISO/IEC 14496-5:2001/Amd.10:2007—Information Technology—Coding of Audio-Visual Objects—Part 5: Reference Software*, ISO/IEC 14496-5:2001/Amd.10:2007, 2010, ISO/MPEG.
[28] "Sound quality assessment material (SQAM) recordings for subjective tests," Eur. Broadcasting Union, 1988, Tech. Rep. 3253.
[29] R. Huber and B. Kollmeier, "PEMO-Q: A new method for objective audio quality assessment using a model of auditory perception," *IEEE Trans. Audio, Speech, Lang. Process.*, vol. 14, no. 6, pp. 1902–1911, Nov. 2006.
[30] "Methods for objective measurements of perceived audio quality," ITU Publication, 2001, ITU-R Rec. BS.1387-1.
[31] "Method for the subjective assessment of intermediate quality level coding systems," ITU Publication, 2001, ITU-R Rec. BS.1534.
[32] D. Miyaguchi, "ABC/Hidden Reference Audio Comparison Tool (Version 1.0)," 2004 [Online]. Available: http://ff123.net/abchr/abchr.html

Stefan Strahl (M'08) received the M.Sc. degree in intelligent systems from Brunel University, London, U.K., in 2000, the Dipl.-Math. degree in mathematics/computer science/medical biology from Leibniz University, Hannover, Germany, in 2003, and the Dr.rer.nat. degree from the University of Oldenburg, Oldenburg, Germany, in 2009.

In 2003, he joined the International Graduate School for Neurosensory Science and Systems, University of Oldenburg, Oldenburg, Germany. In 2007, he was visiting the UCL Ear Institute, London, developing a binaural visual-to-auditory sensory substitution prosthesis. Since 2008, he has been with the Animal Physiology and Behavior Group, University of Oldenburg, developing computational models of the auditory system, and in July 2008 he joined MED-EL GmbH, Austria, working as a Development and Research Engineer on objective measures of the auditory system. His research interests include sparse signal processing, audio coding, multimodal and neurosensory processing, and auditory models.

Heiko Hansen received the diploma degree in physics from the University of Kiel, Kiel, Germany, in 2000, and the Doctor of Science ("rer. nat.") degree from the University of Hamburg, Germany, in 2007.

From 2001 to 2005, he was a Research Assistant at the Max-Planck-Institute for Meteorology, Hamburg, Germany. From 2005 to 2009, he was a Research Assistant at the Department of Physics, Carl von Ossietzky University of Oldenburg, Oldenburg, Germany, and member of the project SFB-TRR 31 "Das aktive Gehör," funded by the German Research Foundation. His research interests are digital signal and audio processing, with special focus on optimal embedded audio coding.

Alfred Mertins (M'95–SM'03) received the Dipl.-Ing. degree in electrical engineering from the University of Paderborn, Paderborn, Germany, in 1984, the Dr.-Ing. degree in electrical engineering and the Dr.-Ing. habil. degree in telecommunications from the Hamburg University of Technology, Hamburg, Germany, in 1991 and 1994, respectively.

From 1986 to 1991, he was with the Hamburg University of Technology, from 1991 to 1995 with the Microelectronics Applications Center Hamburg, from 1996 to 1997 with the University of Kiel, Kiel, Germany, from 1997 to 1998 with the University of Western Australia, and from 1998 to 2003 with the University of Wollongong, Australia. From April 2003 to October 2006, he was with the University of Oldenburg. In November 2006, he joined the University of Lübeck, Lübeck, Germany, as a Professor of Computer Science and Director of the Institute for Signal Processing. His research interests include speech, audio, image, and video processing, wavelets and filter banks, and digital communications.