SCALABLE TO LOSSLESS AUDIO COMPRESSION BASED ON PERCEPTUAL SET PARTITIONING IN HIERARCHICAL TREES (PSPIHT)

Mohammed Raad, Alfred Mertins, and Ian Burnett

School of Electrical, Computer and Telecommunications Engineering
University Of Wollongong, Northfields Ave Wollongong NSW 2522, Australia
email: mr10@uow.edu.au

ABSTRACT

This paper proposes a technique for scalable to lossless audio compression. The scheme presented is perceptually scalable and also provides for lossless compression. It produces smooth objective scalability, in terms of SegSNR, from lossy to lossless compression. The proposal is built around the Perceptual SPIHT algorithm, which is a modification of the SPIHT algorithm and is introduced in this paper. Both objective and subjective results are reported and demonstrate both perceptual and objective measure scalability. The subjective results indicate that the proposed method performs comparably with the MPEG-4 AAC coder at 16, 32 and 64 kbps, yet also achieves a scalable-to-lossless architecture.

1. INTRODUCTION

Lossless audio coding has typically been approached from a signal modeling perspective [1],[2],[3]. The signal is typically modeled using a linear predictor (either FIR or IIR) [2]. The linear predictor decorrelates the audio samples in the time domain and reduces the signal energy requiring coding [1]. The compression ratio of such coders typically depends on the nature of the audio signal being coded with typical values ranging between 1.4 and 5.3 [1].

Scalable audio compression has also been approached from a signal modeling viewpoint. Recent scalable coding schemes, such as the one described in [4], use a composite signal model built on an appropriate combination of Sinusoids, Transients and Noise (STN).

With the increased bandwidth availability in current and future cellular systems, compression schemes that combine both scalability and lossless compression are of interest and potential use. For example, MPEG have started a process of standardization for such a scheme [5]. In this paper, we present a scalable audio coder that allows fine granular scalability (bit-by-bit) as well as competitive compression ratios at the lossless level. The compression scheme is built around transform coding of audio, similar to [6]. In particular, a modified version of the Set Partitioning In Hierarchical Trees (SPIHT) algorithm [7] is used to allow scalability as well as perfect reconstruction. Originally, SPIHT was proposed for image compression, but with a number of modifications it can also be used for audio compression. Our new proposal is named Perceptual SPIHT (PSPIHT). The use of PSPIHT and SPIHT allows the coder to quantize the transform coefficients in such a manner that only the input audio segment's statistics are required; this avoids the design of dedicated entropy code books.

This paper is organized as follows: Section 2 describes the different components of the proposed scalable-to-lossless scheme. Section 3 gives a brief outline of the SPIHT algorithm as well as a complete algorithmic description of PSPIHT. Section 4 presents the lossless and scalable-to-lossless results, and Section 5 provides a brief conclusion.

2. THE SCALABLE TO LOSSLESS SCHEME

The structure of the coder proposed in this paper is depicted in Fig. 1. It consists of the combination of the lossy coder presented in [8], which is based on the Modulated Lapped Transform (MLT) and SPIHT, and a lossless coder for transmitting the error generated by the lossy part. The lossy part is given by the right half of the structure in Fig. 1, and the error coding takes place in the left half. Both parts of the coder are based on the SPIHT algorithm and here we concentrate on the lossy part of the structure, referred to as MLT-PSPIHT.

Referring to Fig.1, the input signal is transformed using a floating point MLT. The transform coefficients are then encoded using PSPIHT, and the bitstream ($bst1$) transmitted to the decoder. Bitstream $bst1$ is further divided into $bst1a$ and $bst1b$ by PSPIHT. This sub-division aims to separate perceptually significant coefficients from perceptually insignificant coefficients such that $bst1a$ contains the perceptually significant coefficients and can thus be transmitted prior to $bst1b$. The encoded $bst1$ is decoded (still at the encoder) and the synthesized audio is then subtracted from the original audio to obtain the output error. Here integer operations are used, such that the error output is integer with a dynamic range typically less than that of the original integer signal. The time-domain error signal is then encoded into bitstream $bst2$, using SPIHT. At the decoder, both bitstreams are received as one global bitstream, with $bst1$ contributing the first part of the total bitstream. The decoder may decode this transform coefficient bitstream at any rate desired; if $bst1$ is exhausted, the decoder recognizes that the remaining bitstream represents the time-domain error signal, which it reconstructs and adds to the synthesized signal.

Fig. 1 highlights that the main components of the compression scheme proposed are PSPIHT and SPIHT. As PSPIHT is a new modification of SPIHT, the next section deals primarily with this algorithm.

3. THE PSPIHT ALGORITHM

PSPIHT is a modification of SPIHT [7] operating in the frequency domain rather than a scale domain. It allows the transmission of perceptually significant coefficients ahead of perceptually insignificant coefficients whilst quantizing both coefficient sets with equal resolution. Such an algorithm maintains the potential for lossless synthesis since energy significant spectral components (that are perceptually insignificant) are not distorted more than perceptually significant spectral components. The primary algorithmic

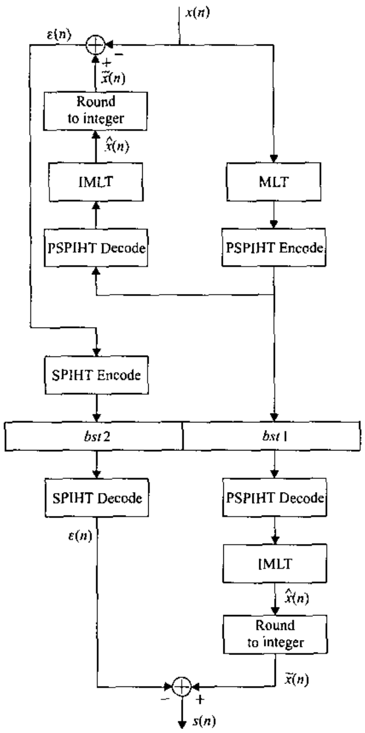

Fig. 1. The scalable-to-lossless scheme based on SPIHT and PSPIHT.

modification focuses on introducing a perceptual significance test to allow the separation of two constituent bitstreams. The perceptual significance test is based on the perceptual entropy of the given coefficient as determined in [9]. For PSPIHT a few new definitions are added to those used by SPIHT and listed in [7]. These are: 1. v_{pe} is a binary vector with perceptual significance information for the sub-band coefficients i.e., if $v_{pe}(n) = 1$ then coefficient n is perceptually significant, otherwise it is perceptually insignificant. 2. LPISP is a list of perceptually insignificant, but energy significant coefficients i.e., LPISP contains pointers to coefficients that are significant in terms of energy (or magnitude) but lie in spectral bands that contain other more significant coefficients which have masked them. 3. Finally, we denote the perceptually significant component of $bst1$ as $bst1a$ and the perceptually insignificant component as $bst1b$. Fixed limits can be set for the sizes of $bst1a$ and $bst1b$.

The complete algorithm is listed below. Here, the original matrix notation of SPIHT has been dropped in favour of vector notation which is more applicable to audio signals. The offspring of a coefficient are determined according to the following equation, where N is the number of offspring used:

$$O(i) = iN + \{0, N-1\}. \qquad (1)$$

In our case, $N = 4$ is chosen. This method of choosing the offspring does not provide the same type of parent-offspring behaviour seen in wavelet based hierarchical trees, however in practice we have found it to be a good choice for MLT based coding.

In the sorting pass, the energy significance test is maintained as the first test. Sorting bits are sent to $bst1a$ until an energy significant coefficient is encountered. This coefficient is then tested for perceptual significance by checking the corresponding entry in v_{pe}. If the coefficient is found to be significant (and $bst1a$ is not full), the sign bit and further refinement bits are sent to $bst1a$, otherwise these bits are sent to $bst1b$. The perceptual significance test is only applied to individual coefficients and not to whole sets as in the case of the energy significance test. The same process is followed at the decoder which obtains the test results (the sorting information), sign bits and significant bits from the bitstream. Note that the major task of the algorithm is to arrange the generated bitstream so that it reflects perceptual significance, allowing more perceptually accurate synthesis at lower rates. Some extra overhead is encountered in the bitstream formatting as a pointer must also be transmitted indicating the length of $bst1a$. This is necessary for the decoder to be able to divide the total bitstream correctly and to allow $bst1a$ to be less than its hard-coded maximum length, if the signal contains fewer significant components than expected. Although the listed algorithm outputs perceptual significance information it does so only for energy significant components and, even then, only when there is space in $bst1a$. Hence, it would be rare to encounter a situation where all of v_{pe} is transmitted.

Algorithm PSPIHT:

```
1) Initialization: output $n = \lfloor log_2(max_i|c_i|) \rfloor$;
    set the LSP as an empty list, and add the coordinates
    $(i) \in H$ to the LIP, and only those with descendants
    also to the LIS, as type $A$ entries.
    Set LPISP as an empty set.
2) Sorting Pass:
    2.1) for each entry $i$ in the LIP do:
        If $bst1a$ is not full
            2.1.1) output $S_n(i)$ to $bst1a$
        Else
            2.1.1) output $S_n(i)$ to $bst1b$
        2.1.2) if $S_n(i) = 1$ and $bst1a$ is not full then
            output $v_{pe}(i)$ to $bst1a$
            else if $bst1a$ is full
                move $(i)$ to LPISP
                and output the sign of $c_i$ to $bst1b$;
                If $v_{pe}(i) = 1$ then move $(i)$ to the LSP
                    If $bst1a$ is not full
                    output the sign of $c_i$ to $bst1a$;
                    Else output the sign of $c_i$ to $bst1b$;
                Else move $(i)$ to the LPISP and output
                the sign of $c_i$ to $bst1b$
    2.2) for each entry $(i)$ in the LIS do:
        2.2.1) if the entry is of type $A$ then
            If $bst1a$ is not full
                • output $S_n(D(i))$ to $bst1a$
            Else
                • output $S_n(D(i))$ to $bst1b$
            • if $S_n(D(i)) = 1$ then
                * for each $(k) \in O(i)$ do:
                    If $bst1a$ is not full
                        • output $S_n(k)$ to $bst1a$
                    Else
                        • output $S_n(k)$ to $bst1b$
                    • if $S_n(k) = 1$
                        If $bst1a$ is not full
                        output $v_{pe}(k)$ to $bst1a$
                        Else do not output $v_{pe}(k)$,
            move $(k)$ to the LPISP
            and output the sign of $c_i$ to $bst1b$
                If $v_{pe}(k) = 1$ then
                add $(k)$ to the LSP
                    If $bst1a$ is not full
                    output the sign of $c_k$ to $bst1a$
                    Else output the sign of $c_k$ to $bst1b$
                Else add $(k)$ to the LPISP and
                output the sign of $c_k$ to $bst1b$
        •   if $S_n(k) = 0$ then add $(k)$ to the
            end of the LIP;
    *   if $L(i) \neq \emptyset$ then move $(i)$ to the
        end of the LIS as an entry of type $B$,
        and go to Step 2.2.2); otherwise, remove
        entry $(i)$ from the LIS;
2.2.2)  if the entry is of type $B$ then
    If $bst1a$ is not full
    •   output $S_n(L(i))$ to $bst1a$
    •   output $S_n(L(i))$ to $bst1b$
•   if $S_n(L(i)) = 1$ then
    *   add each $(k) \in O(i)$ to the end of
        the LIS as an entry of type $A$;
    *   remove $(i)$ from the LIS.
3) Refinement Pass: for each entry $(i)$ in the LSP,
   except those included in the last sorting pass (i.e., with
   same $n$), output the $n$th most significant bit of $|c_i|$
   to $bst1a$ if it is not full,
   otherwise output the $n$th most significant bit of $|c_i|$ to $bst1b$
   for each entry $(i)$ in the LPISP,
   except those included in the last sorting pass (i.e., with
   same $n$), output the $n$th most significant bit of $|c_i|$ to $bst1b$
4) Quantization-Step Update: set $n := n - 1$ and go to step 2.
```

4. RESULTS

Objective and subjective results are presented in this section. The objective results illustrate the objective scalability of the coder and the lossless compression performance of the proposed scheme. The subjective results compare the lossy part of the proposed scheme to the MPEG-4 AAC coder at 16, 32 and 64 kbps. The synthesized audio of PSPIHT based schemes at these rates is also compared subjectively to illustrate the subjective scalability. The implementation used to obtain the presented results limits $bst1$ to 96 kbps, and $bst1a$ to 64 kbps. The MLT coefficients are quantized using an 18 bit uniform quantizer. The test material used is listed in Table 1.

4.1. Objective results

Fig. 2 shows the Segmental Signal-to-Noise-Ratio (SegSNR) for test file x1. The objective scalability of the proposed scheme is clear from the figure. It is notable that a knee point begins to appear at around the 96 kbps mark (which is the limit set for $bst1$) and that the rate of increase in SegSNR is greatest for bit rates below 64 kbps. However, the results shown must be put in perspective. In Fig. 2, the results shown for bit rates above 96 kbps are for frames that have not been perfectly reconstructed, the number of which decreases with increasing bit rate.

Table 2 lists the lossless compression results for the proposed scheme. The mean compression ratio obtained across all of the test material was 2.29, with the greatest compression obtained when harmonic signals (such as x2) are coded. The state of the art lossless compression algorithms produce compression ratios that vary, according to signal content, between 1.4 and 5.3 [1]. Table 2 shows that the proposed lossless compression scheme's performance is competitive with the state of the art. The advantage of the proposed scheme is the ability to recover both a lossy and lossless representation of the original signal from the same bit stream, depending on bandwidth availability.

Table 1. The Signal Content.

Signal	Signal Content	Signal	Signal Content
x1	Bass	x9	English Female Speech
x2	Electronic Tune	x10	French Female Speech
x3	Glockenspiel	x11	German Female Speech
x4	Glockenspiel	x12	English Male Speech
x5	Harpsicord	x13	French Male Speech
x6	Horn	x14	German Male Speech
x7	Quartet	x15	Trumpet
x8	Soprano	x16	Violoncello

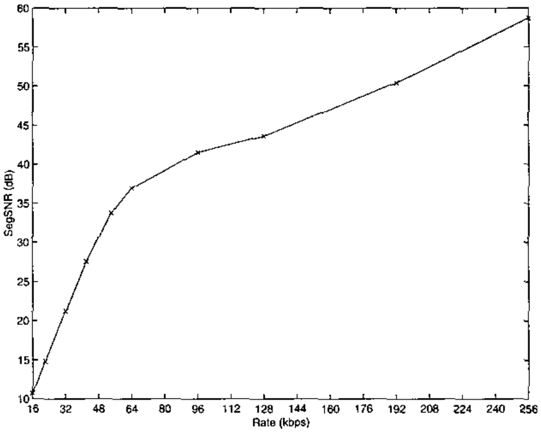

Fig. 2. An example of the SegSNR results obtained using the proposed scheme (using file x1). For rates below 96 kbps the SegSNR was computed for all frames, while above 96 kbps only the imperfectly reconstructed ones were used.

4.2. Subjective Results

Listening tests have been conducted to compare the proposed scheme to the MPEG-4 AAC coder so as to obtain an idea of the perceptual scalability of the MLT-PSPIHT scheme. A total of 39 subjects were asked to listen to two differently coded versions of the same sound (labelled A and B) and indicate which version was more preferable. A scale of 1 to 5 was used as shown in Table 3. The comparison tests involved comparing the AAC coder and the PSPIHT coder at 16, 32 and 64 kbps. The PSPIHT coder at lower rates has also been tested against the PSPIHT coder at higher rates. Specifically, the PSPIHT coder at 16 kbps was compared to the PSPIHT coder at 32 kbps and the latter compared to the PSPIHT coder at 64 kbps.

The mean scores for all the comparisons have been obtained as well as the differences between the mean scores and the value 3. Fig. 3 shows the results with their 95% confidence intervals. In the results shown in Fig. 3 a score that is below zero indicates that the first coder mentioned in the label is better than the second one according to the scale used in the test. For example, the first data

Table 2. Results for the Lossless PSPIHT Coder.

Signal	Mean Rate (kbps)	Compression Ratio	Bits/Sample
x1	349	2.02	7.91
x2	134	5.33	3.03
x3	224	3.15	5.08
x4	305	2.32	6.92
x5	393	1.80	8.91
x6	244	2.90	5.53
x7	392	1.8	8.89
x8	347	2.03	7.87
x9	405	1.74	9.18
x10	422	1.67	9.57
x11	385	1.84	8.73
x12	396	1.78	8.98
x13	376	1.88	8.53
x14	387	1.82	8.78
x15	296	2.39	6.71
x16	331	2.13	7.51

Table 3. The subjective comparison scale used.

Score	Subjective opinion
1	A much better than B
2	A better than B
3	A and B are the same
4	A worse than B
5	A much worse than B

point plotted is below zero indicating that, on average, the AAC coder is better than the PSPIHT coder (both at 16 kbps) within the shown 95% confidence interval.

The subjective results shown in Fig. 3 clearly show the perceptual scalability of the MLT-PSPIHT scheme. There is a distinct difference between coding at 16 and 32 kbps, and a clear (although reduced) difference between 32 and 64 kbps. The comparison between the AAC and the MLT-PSPIHT coder shows that both coders perform quite similarly at 16 and 32 kbps (with 95% confidence). The AAC outperforms the MLT-PSPIHT scheme at 64 kbps by a margin of 0.49 indicating a small overall difference in quality. However, the AAC coder has been used as a fixed rate coder for the given rates and does not allow for any bit rate scalability. The MLT-PSPIHT coder, on the other hand, produces an embedded bitstream that is bit-by-bit scalable and can be easily adapted, in a transcoding stage, to any bit rate requirement by simply leaving out parts of the encoded bitstream. For example, a 64 kbps bitstream can be turned into a 32 kbps one by removing the second halves of the sets of bits belonging to the various transform blocks. Moreover, the complete MLT-PSPIHT coder of Fig. 1 scales up to lossless, provided the allowed bit rate is sufficiently high.

5. CONCLUSION

We have presented a scalable to lossless compression scheme. It is based on the PSPIHT and SPIHT algorithms as well as the MLT. The PSPIHT algorithm is a modified version of the SPIHT algorithm that allows the transmission of perceptually significant coefficients prior to perceptually insignificant ones. Our objective results show that the scheme scales smoothly and objectively (in terms of SegSNR) from lossy to lossless compression. The lossless compression obtained is competitive with the state of the art, although, like all lossless compression schemes, the compression ratio depends on the signal content. The subjective listening tests conducted show the perceptual scalability of the PSPIHT scheme as well as its comparable performance with the MPEG-4 AAC coder. The main advantage of the PSPIHT scheme is that, given an encoded bit stream, the synthesized audio for any lower bit rate is easily obtainable. This is a very useful property for variable rate transmission and transcoding.

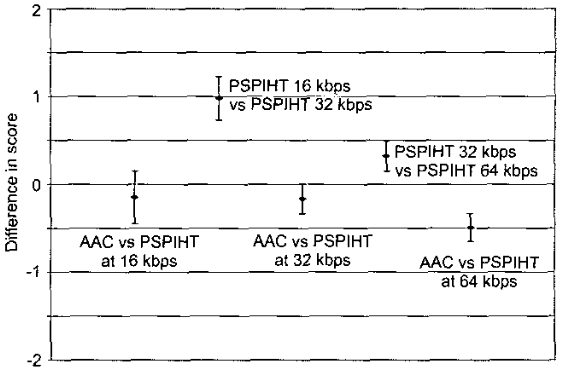

Fig. 3. Listening test results.

6. ACKNOWLEDGMENTS

Mohammed Raad is a recipient of an Australian Postgraduate Award of Industry (APAI) grant. This work is supported by the Motorola Australian Research Centre. The authors wish to thank Ms. Melanie Jackson for her help in conducting the listening tests.

7. REFERENCES

[1] M. Hans and R.W. Schafer, "Lossless compression of digital audio," *IEEE Signal Processing magazine*, vol. 18, no. 4, pp. 21–32, July 2001.

[2] P.G. Craven and M.J. Law, "Lossless compression using IIR prediction filters," AES 102nd convention, AES preprint 4415, March 1997.

[3] A.A.M.L. Bruekers, W.J. Oomen, and R.J. van der Vleuten, "Lossless coding for DVD audio," AES 101st convention, AES preprint 4358, November 1996.

[4] T.S. Verma, *A perceptually based audio signal model with application to scalable audio compression*, Ph.D. thesis, Department of Electrical Engineering, Stanford university, October 1999.

[5] T. Moriya, "Report of AHG on issues in lossless audio coding," ISO/IEC JTC1/SC29/WG11 M7955, March 2002.

[6] T. Liebchen, M. Purat, and P. Noll, "Lossless transform coding of audio signals," *Proceedings of the 102nd AES convention, AES preprint 4414*, March 1997.

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

[8] M. Raad, A. Mertins, and I. Burnett, "Audio compression using the MLT and SPIHT," *Proceedings of DSPCS' 02*, pp. 128–132, 2002.

[9] J.D. Johnston, "Estimation of perceptual entropy using noise masking criteria," *Proceedings of ICASSP-88*, vol. 5, pp. 2524–2527, 1988.