JOINT ENTROPY-SCALABLE CODING OF AUDIO SIGNALS

Mahmood Movassagh *Joachim Thiemann* *Peter Kabal*

Department of Electrical and Computer Engineering
McGill University, Montreal, Canada
Email: {mahmood.movassagh@mail.mcgill.ca, joachim.thiemann@mcgill.ca, peter.kabal@mcgill.ca}

ABSTRACT

A fine grain scalable coding for audio signals is proposed where the entropy coding of the quantizer outputs is made scalable. By constructing a Huffman-like coding tree where internal nodes can be mapped to reconstruction points, we can prune the tree to control the distortion of the quantizer. Our results show the proposed method improves existing similar work and significantly outperforms scalable coding based on reconstruction error quantization as used in practical systems, eg. MPEG-4 audio.

Index Terms— Scalable coding, Entropy coding, Quantization

1. INTRODUCTION

Bit-rate scalability has been a necessary requirement in multimedia communications. Without the need to re-encode the original signal, it allows for improving the quality of an audio/video signal as more of a total bit stream becomes available, or lowering the quality if channel condition deteriorates. Scalability can also provide robustness to packet loss for transmission over packet networks. In such systems, very robust channel coding can be performed for the core bitstream so that all the receivers can receive it without loss. The rest of the bitstream is sent with normal channel coding. Thus, if the packets are lost, the signal can be still reconstructed at base level quality.

Several scalable coding systems have been proposed so far, including using wavelet transforms [1], bit-plane based coding [2, 3], and fine-grain scalable coding [4, 5]. One popular scalable coding system, at the core of AAC scalable coding [6], is a system based on reconstruction error quantization (REQ). In REQ (Fig. 1), the signal is quantized by an optimal quantizer designed for a minimum bit rate and acceptable distortion (the base layer). Enhancement layers improve the quality of the base layer signal, refining the quantization by subtracting the quantized signal from the original. This error signal is quantized, encoded and transmitted as the first enhancement layer. This enhancement step can be repeated, to form an ordered set of layers. From the base up, each additional layer that the receiver receives is used to refine the quality of the decoded signal.

In terms of Rate-Distortion (RD) performance, REQ is optimal for the Mean Square Error (MSE) criterion. It asymptotically achieves the performance of an equivalent non-scalable coding system [7] if the rate is measured by the entropy of resulting output symbols. However, in practical coding systems symbols need to be encoded in a bitstream using an entropy coding scheme, which adds an overhead for each layer.

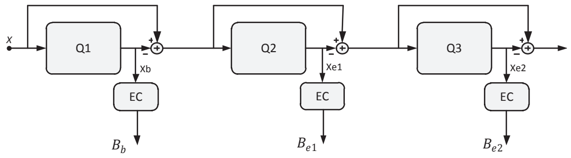


Fig. 1. Salable Audio Coding based on REQ

In the example system in Fig. 1, if Huffman coding is performed separately for each layer output, the upper bound of the bitrate for each layer is the entropy of the symbols plus one bit; thus the upper bound of the combined layers is the entropy of all symbols combined plus N bits, where N is the number of layers.

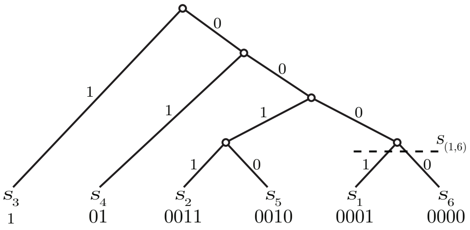
Scalable entropy coding was proposed in [8] assuming a general set of random variables representing quantizer reconstruction points. In this paper, we show that it is important to consider the positions of the reconstruction points within the quantizer intervals, and discuss Rate-Distortion issues. A new set of equations are derived for the general case where the reconstruction points can take any arbitrary positions within the quantizer intervals. This joint entropy-scalable coding (JESC) scheme is applied to the quantizer used by the AAC codec, and we compare JESC to REQ scalable coding and the progressive entropy coding (PEC) scheme of [8].

2. JOINT ENTROPY-SCALABLE CODING

Huffman coding is a very common entropy coding technique. The codebook is generated by assigning symbols to leaves on an unbalanced binary tree (Fig. 2). To build the coding tree nodes are created by joining either leaves or nodes with lowest probability until all leaves are part of a single tree.

Suppose we want to reduce the bitrate of the information of the stream of symbols. We can do so by pruning the tree, for example at node $s_{(1,6)}$ in Fig. 2: in the resulting bitstream, the code 000 will then appear with probability 0.04. However, in the context of the original system, this codeword has no meaning since it cannot be mapped to any symbol.

978-1-4673-0046-9/12/$26.00 ©2012 IEEE



symbol	p_q
s_1	.02
s_2	.03
s_3	.8
s_4	.1
s_5	.03
s_6	.02

Fig. 2. Huffman coding tree

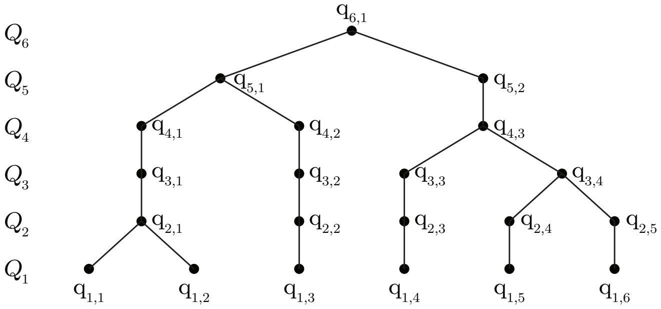


Fig. 3. Creating new quantizers by merging the nodes

2.1. Relating internal nodes of a Huffman tree to quantization reconstruction points

Now consider the case where symbols represent the outputs of a scalar quantizer. In the construction of a regular Huffman tree we just search for the two reconstruction points of the smallest probabilities to make a new node. However, if we ensure also that at a joining step the leaves represent quantizer outputs that are neighbouring Voronoi regions, the resulting node can be assigned a new reconstruction point and be treated as a leaf. Thus, we can effectively get a new quantizer if the tree is pruned at that node. Such a set of quantizers is shown in Fig. 3, where the tree describes a set of quantizers ($Q_1, Q_2, ...$) resulting from pruning a quantizer-encoding tree from the bottom up.

As the tree gets pruned, each new quantizer has a smaller entropy and larger distortion compared to the previous one. The reduction of the average bit rate of the quantizer is obtained by

$$\Delta B = w_1 b + w_2 b - (w_1 + w_2)(b-1) = w_1 + w_2 = W \quad (1)$$

where b is the number of bits assigned to two nodes before merging and w_1 and w_2 the probabilities of the nodes or leaves. By pruning the tree at different possible nodes, we can create a large set of quantizers and hence obtain fine grain bit rate scalability. We note that the receiver needs to know which tree to use to decode a given bitstream; thus, a number indicating the pruning level (that is, the quantizer index, $Q_1, Q_2, \ldots$) needs to be sent as side information. This side information is nothing extra compared to the practical scalable coders (including REQ) where a scale factor is sent for each layer so that the receiver can know which quantization resolution is used for them.

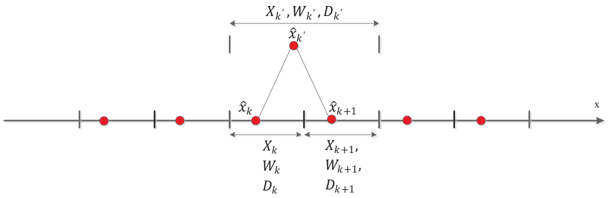


Fig. 4. Merging two reconstruction points in a quantizer

2.2. Merging quantizer regions to build the coding tree

The construction of the scalable entropy coding tree is determined not only by the probability density function (*pdf*) of the signal to be encoded, but also the distortion measure we wish to optimize to. For a signal x with pdf given by $f(x)$, consider the scalar quantizer $Q(x)$ with distortion

$$\begin{aligned} D = E[e^2] &= \int_x (x - Q(x))^2 f(x)\, dx \\ &= \sum_i W_i D_i, \end{aligned} \quad (2)$$

where D_i is the conditional distortion in each interval i of the quantizer. W_i is the probability of $x \in X_i$, so if we write $\hat{x}_i = Q(x)|_{x \in X_i}$,

$$D_i = E[e^2 | X_i] = \int_{x \in X_i} (x - \hat{x}_i)^2 \frac{f(x)}{W_i} dx. \quad (3)$$

To find the difference in distortion for a quantizer that has regions merged as described above, suppose we merge the adjacent quantizer regions X_k and X_{k+1}. Now, we have a new quantizer with slightly higher distortion, with distortion given by

$$\begin{aligned} D' &= \sum_{i \neq k, k+1} W_i D_i + W_{k'} D_{k'} \\ &= D - (W_k D_k + W_{k+1} D_{k+1}) + W_{k'} D_{k'}. \end{aligned} \quad (4)$$

The difference in distortion between the old and new quantizers can now be written as $\Delta D = D' - D$ or

$$\Delta D = W_{k'} D_{k'} - (W_k D_k + W_{k+1} D_{k+1}). \quad (5)$$

Now suppose we want to merge two quantization regions to form a new interval (See Fig. 4). Changing the reconstruction point of one interval does not change the distortions of other intervals, so the best reconstruction point for the new node is obtained by minimizing the conditional distortion in the new node's interval,

$$\begin{aligned} D_{k'} &= E[(x - \hat{x}_{k'})^2 \mid X_{k'}] \\ &= \int_{x_k}^{x_{k+1}} (x - \hat{x}_{k'})^2 \frac{f(x)}{W_{k'}}\, dx, \end{aligned} \quad (6)$$

giving

$$\begin{aligned} \frac{dD_{k'}}{d\hat{x}_{k'}} &= -2 \int\limits_{x\in X_{k'}} (x-\hat{x}_{k'})\frac{f(x)}{W_{k'}}\,dx = 0 \\ \Rightarrow \hat{x}_{k'} &= \int\limits_{x\in X_{k'}} x\frac{f(x)}{W_{k'}}\,dx \\ &= E[x \mid X_{k'}]. \end{aligned} \tag{7}$$

The new distortion $D_{k'}$ can be expressed in terms of the conditional expectations of the two merging nodes in their own intervals. Thus,

$$\begin{aligned} W_{k'}D_{k'} &= W_{k'}E[e^2 \mid X_{k'}] \\ &= W_kE[e^2 \mid X_k] + W_{k+1}E[e^2 \mid X_{k+1}], \end{aligned} \tag{8}$$

where

$$\begin{aligned} E[e^2 \mid X_k] &= \int\limits_{x\in X_k} (x-\hat{x}_{k'})^2\frac{f(x)}{W_k}\,dx \\ &= \int\limits_{x\in X_k} [(x-\hat{x}_k)^2 + (\hat{x}_k-\hat{x}_{k'})^2 \\ &\quad + 2(x-\hat{x}_k)(\hat{x}_k-\hat{x}_{k'})]\frac{f(x)}{W_k}\,dx \\ &= D_k + (\hat{x}_k-\hat{x}_{k'})^2 \\ &\quad + 2(\hat{x}_k-\hat{x}_{k'})(E[x \mid X_k]-\hat{x}_k), \end{aligned} \tag{9}$$

and

$$\begin{aligned} E[e^2 \mid X_{k+1}] &= D_{k+1} + (\hat{x}_{k+1}-\hat{x}_{k'})^2 \\ &\quad + 2(\hat{x}_{k+1}-\hat{x}_{k'})(E[x \mid X_{k+1}]-\hat{x}_{k+1}). \end{aligned} \tag{10}$$

Consequently,

$$\begin{aligned} W_{k'}D_{k'} &= W_kE[e^2 \mid X_k] + W_{k+1}E[e^2 \mid X_{k+1}] \\ &= W_kD_k + W_{k+1}D_{k+1} \\ &\quad + W_k(\hat{x}_k-\hat{x}_{k'})^2 + W_{k+1}(\hat{x}_{k+1}-\hat{x}_{k'})^2 \\ &\quad + 2W_k(\hat{x}_k-\hat{x}_{k'})(E[x \mid X_k]-\hat{x}_k) \\ &\quad + 2W_{k+1}(\hat{x}_{k+1}-\hat{x}_{k'})(E[x \mid X_{k+1}]-\hat{x}_{k+1}), \end{aligned} \tag{11}$$

which gives us the distortion of the new interval in terms of the new and old reconstruction points and the weights and conditional expectations of the old nodes.

Finally, using (5) we get

$$\begin{aligned} \Delta D &= W_k(\hat{x}_k-\hat{x}_{k'})^2 + W_{k+1}(\hat{x}_{k+1}-\hat{x}_{k'})^2 \\ &\quad + 2W_k(\hat{x}_k-\hat{x}_{k'})(\bar{x}_k-\hat{x}_k) \\ &\quad + 2W_{k+1}(\hat{x}_{k+1}-\hat{x}_{k'})(\bar{x}_{k+1}-\hat{x}_{k+1}), \end{aligned} \tag{12}$$

where the conditional expectations have been replaced by $\bar{x}_k$ and $\bar{x}_{k+1}$.

Equation (12) gives the distortion increase for a general case where the reconstruction points can be at arbitrary positions within quantizer intervals. The new reconstruction point $\hat{x}$ in this equation can be obtained as

$$\hat{x}_{k'} = E[x \mid X_{k'}] = \frac{W_k}{W_{k'}}\bar{x}_k + \frac{W_{k+1}}{W_{k'}}\bar{x}_{k+1}. \tag{13}$$

Now that we have equations giving distortion increase and bit rate decrease resulting from merging, we need a measure for finding the best choice of nodes for merging at each step. In Huffman coding we pick the two nodes which have the smallest probabilities. Here we need a measure which considers both weights (probabilities) and the distortion increase resulted from merging. The measure we used is

$$M = \alpha\Delta D^2 + \beta\Delta B^2 = \alpha\Delta D^2 + \beta\Delta W^2, \tag{14}$$

similar to [8]. The weighting parameters α and β were set empirically to values which gave the best results with $\alpha = \beta = 0.5$.

In the following section we discuss a specific case where a Uniform Threshold Quantizer (UTQ) is used with the JESC.

3. JESC WITH THE AAC QUANTIZER

The quantization formula which is used in AAC [9] is given by

$$\begin{aligned} i_x &= \text{sgn(x)} * \text{nint}(\Delta|x|^{0.75} - 0.0946) \\ \hat{x} &= \text{sgn(x)} * (\frac{|i_x|}{\Delta})^{\frac{4}{3}}, \end{aligned} \tag{15}$$

where Δ is the quantizer step size (or scale factor) parameter, and nint() and sgn() denote the nearest integer and signum function. This quantization formula is based on the assumption that the input signal is Laplacian. The optimal quantizer for exponential and Laplacian signals, a special case of exponential signals, is the UTQ with a dead-zone around zero [10]. In such a quantizer, reconstruction points are not in the middle of the quantizer intervals and are dependent on the statistics of the input signal. However, quantization formula in AAC uses the ratio of the offset to the step size of the quantizer, which remains the same (0.0946) as the step parameter Δ changes.

Consider a Laplacian signal and let us define r as the ratio of the offset α to the width of the quantizer intervals[1] d, $r = \frac{\alpha}{d}$. The offset α is defined as the distance between the reconstruction point and the lower threshold of the intervals. Also, let $k = \frac{b}{d}$ where b is the scale parameter of the Laplacian distributed signal with zero mean. Then, it can be shown that r is obtained by

$$r = k - \frac{1}{e^{\frac{1}{k}} - 1}. \tag{16}$$

Changing the parameter Δ in the AAC quantization formula is equivalent to changing the scale parameter of the Laplacian input signal. Thus, when Δ is not equal to 1 we need to replace k with $k\Delta$ in the above equation, so we get

$$r = k\Delta - \frac{1}{e^{\frac{1}{k\Delta}} - 1}. \tag{17}$$

This relationship of r_Δ versus step size parameter Δ is shown in Fig. 5 for $k = 1$. It can bee seen that the ratio r is neither constant nor has a linear relation with Δ. Thus (12) provides a good general distortion increase measure for all types of quantizers where the reconstruction points can take any positions within the intervals.

[1] Excluding the dead-zone area where the reconstruction point is in the middle, i.e. zero.

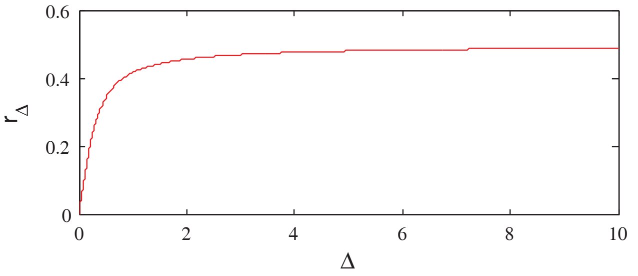


Fig. 5. Offset to interval width ratio r versus step size parameter Δ ($k = 1$) for AAC quantizer.

4. SIMULATION RESULTS

We compare a 4 layer REQ to our JESC scalable coders. We assume a 10 bit full quantizer for our JESC system and a set of quantizers using 3, 2, 2, 2 bits respectively for the base and the 3 enhancement layers of the REQ system.

As in [7] we assume $\sigma^2 = 100$. For REQ, each layer quantization uses the AAC quantization formula and Huffman coding on the quantizer outputs. Figure 6 shows the comparison between fine grain JESC, the progressive entropy coding (PEC) of [8], and the REQ scalable coding in terms of bitrate-distortion performance. The 4 rate-distortion pairs shown by points with dashed bars show the four possible rates of the REQ scheme, that is, base layer + one enhancement layer, base layer + two enhancement layers and so on. It can be seen in average there is a difference of about 4dB between the JESC and REQ. A one layer REQ performs slightly better than JESC since in this case the Huffman coding applied directly to the base layer is more efficient, however JESC becomes more efficient as more number of layers are used. This shows the REQ becomes more sub-optimal for rate-distortion performance as the number of layers is increased. It can also be seen that JESC is performing clearly better than PEC.

5. CONCLUSION

REQ scalable coding is a practical scalable coding scheme which is used in MPEG-4 audio coding. In such a coder the entropy coding is performed separately for each layer and the coder becomes suboptimal as the number of layers increases. We propose a scalable coding method for a single quantizer in which the encoding is performed in a fine grain manner. The proposed method considerably outperforms REQ scalable coding and is a suitable replacement for practical scalable audio coders. In early future we will consider applying JESC to VQ as well.

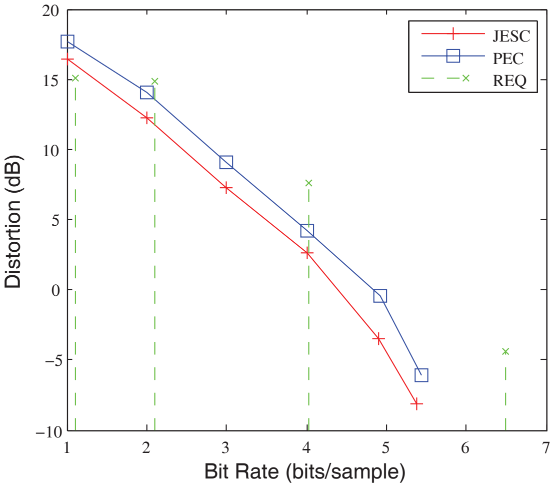


Fig. 6. BitRate-Distortion comparison between JESC PEC and REQ scalable coders. The two curve are the bitrate-distortion plots for the fine grain JESC and PEC scalable coders and the points with dashed bars show the 4 bitrate-distortion pairs resulting from the 4 layers in REQ scalable coder. The input signal for all coders was Laplacian with zero mean and $\sigma^2 = 100$.

6. REFERENCES

[1] D. Ning and M. Deriche, “A bitstream scalable audio coder using a hybrid WLPC-wavelet representation,” in *Proc. IEEE ICASSP*, vol. 5, pp. V–417–20, Apr. 2003.

[2] H. Huang, H. Shu, and S. Rahardja, “Bit-plane arithmetic coding for Laplacian source,” in *Proc. IEEE ICASSP*, pp. 3358 –3361, Mar. 2010.

[3] D. H. Kim, J. H. Kim, and S. W. Kim, “Scalable lossless audio coding based on MPEG-4 BSAC,” in *Proc. 113th AES Conv.*, Paper Number:5679, Oct. 2002.

[4] S. Kim, S. Park, and Y. B. Kim, “Fine grain scalability in MPEG-4 Audio,” in *Proc. 111th AES Conv.*, Paper Number:5491, Nov. 2001.

[5] R. Yu, S. Rahardja, L. Xiao, and C. C. Ko, “A fine granular scalable to lossless audio coder,” *IEEE Trans. Audio, Speech, Lang. Process.*, vol. 14, pp. 1352 –1363, Jul. 2006.

[6] R. Geiger, J. Herre, S. Kim, X. Lin, S. Rahardja, M. Schmidt, and R. Yu, “ISO/IEC MPEG-4 high-definition scalable advanced audio coding,” in *Proc. 120th AES Conv.*, Paper Number:6791, May 2006.

[7] A. Aggarwal, S. L. Regunathan, and K. Rose, “Efficient bitrate scalability for weighted squared error optimization in audio coding,” *IEEE Trans. Audio, Speech, Lang. Process.*, vol. 14, pp. 1313 –1327, Jul. 2006.

[8] T. Verma and T. Meng, “A scalable entropy code,” in *Proc. IEEE Data Compression Conf. (abstract in the IEEE database, for full paper use other sources)*, p. 581, Mar. - Apr. 1998.

[9] M. Bosi, K. Brandenburg, S. Quackenbush, L. Fielder, K. Akagiri, H. Fuchs, M. Dietz, J. Herre, G. Davidson, and Y. Oikawa, “ISO/IEC MPEG-2 advanced audio coding,” *J. Audio Eng. Soc.*, vol. 45, pp. 789–814, Oct. 1997.

[10] G. J. Sullivan, “Efficient scalar quantization of exponential and laplacian random variables,” *IEEE Trans. Inform. Theory*, vol. 42, pp. 1365 –1374, Sep. 1996.