VARIABLE-SIZE VECTOR ENTROPY CODING OF SPEECH AND AUDIO

Yair Shoham

Bell Laboratories, Lucent Technologies
700 Mountain Ave.
Murray Hill, NJ 07974, USA

ABSTRACT

Many modern analog media coders employ some form of entropy coding (EC). Usually, a simple per-letter EC is used to keep the coder's complexity and price low. In some coders, individual symbols are grouped into small fixed-size vectors before EC is applied. In this work we extend this approach to form Variable-Size Vector EC (VSVEC) in which vector sizes may be from 1 to several hundreds. The method is, however, complexity-constrained in the sense that the vector size is always as large as allowed by a pre-set complexity limit. The idea is studied in the framework of an MDCT transform coder. It is shown experimentally, using diverse audio material, that a rate reduction of about 37% can be achieved. The method is, however, not specific to MDCT coding but can be incorporated in various speech, audio, image and video coders.

1. INTRODUCTION

Analog media (speech, audio, image, etc) source coding is often carried out in two steps. First, a *lossy* transformation of the analog data into discrete symbols is performed. Second, *lossless* compression, commonly referred to as entropy coding, is performed to further reduce the bit stream length. Often, the distinction between these two steps is vague or impossible (example: Entropy-based vector quantization [8]). Sometimes, one step is entirely missing, as in most standard speech coders where no entropy coding is used (see ITU-T standards G.728, G.729, etc) or in the opposite case of file compression (Ziv-Lempel and derivatives [9]) where no lossy coding is used.

In simple EC applications, the source symbols are processed individually (per-letter EC) using techniques like Huffman coding [1], [2] and arithmetic coding [3], [4]. More complex EC coders parse the symbol sequence into fixed or variable-size strings to form Vector EC (VEC). [9] [5]. VEC utilizes inter-symbol dependencies to achieve high compression ratios because, fundamentally, the entropy of the combined symbols is never greater than that of the elementary symbols and most of the times it is significantly lower. The price paid for this is an increase in complexity which usually grows exponentially with the vector size and quickly becomes unmanageable. Also, the VEC coder usually requires a considerable "lookahead". This translates into coding delay that may be disadvantageous in communication applications. Real-time coders usually use per-letter EC for speed, simplicity and efficiency [6], [7]. However, in some modern audio coders, an attempt is made to employ a simple form of VEC with fixed vector size of 2 to 5. Examples are the PAC [11], AAC [12] and the PTC [13] coders.

In this work, an improved VEC is proposed. The proposed algorithm is $size-unconstrained$ but $complexity-constrained$. The "unbounded" vector size changes as a function of the local statistics and becomes part of the information stream. Subject to allowed complexity, the vector size may be as small as 1 and as big as the whole frame, a few hundreds or even a few thousands symbols (depending on the frame size). We refer to this approach as variable-size VEC (VSVEC). The proposed VSVEC is studied in the framework of a transform coder that employs the commonly used Modified Discrete Cosine Transform (MDCT) [10]. The coder performs lossy coding of the MDCT coefficients using a set of scalar quantizers (or possibly, a set of low-dimensional vector quantizers). The quantizer resolution is dynamically adjusted by a perceptual control unit (PCU) for best perceptual performance. The symbols emitted from these quantizers are subjected to the VSVEC unit for further lossless compression. In the next section, the VSVEC algorithm is described in details.

2. VARIABLE-SIZE VECTOR ENTROPY CODING

2.1. Introduction

Let S_n be a discrete symbol, typically, a quantizer index. It is assumed (without loss of generality) to be a non-negative integer in the range $0,..,\mathbf{S}-1$. The index n is the location of the symbol in a frame of size N. The first step in the VSVEC process is the grouping of the symbols S_n into variable-size vectors represented by a new set of *combined symbols* (vectors) C_m of size N_m, where the index m points to the mth such symbol in the current frame. The size N_m may take any value, up to the frame size N. In this study we allow only contiguous and non-overlapping grouping. Otherwise, the segmentation is unconstrained. Non-contiguous grouping may be explored in future work on VSVEC. It is immediately evident that the EC task is enormously complex since the alphabet of C_m is astronomic and so is the number of possible segmentations. For the VSVEC to be practical, some structure has to be introduced to the segmentation process and to the way C_m are generated. This is discussed in the next subsection.

2.2. Segmentation

A segmentation is a set of indices $k_m, m = 0, .., M-1$. M is the number of segments. k_m is an address to the $m-th$ subframe whose size is N_m. Therefore, $k_m = \sum_{i=0}^{m-1} N_i < N$. The $m-th$ subframe is associated with a symbol C_m which is encoded to produce a binary word of b_m bits. An optimal segmentation $\{k_m\}_{opt}$ is defined as the one that minimizes the total length of the output binary stream associated with the current frame:

$$\{k_m\}_{opt} = \arg \min_{all\ segmentations} \sum_{i=0}^{i=M-1} b_i \qquad (1)$$

Segmentation of symbol sequences subject to an optimality criterion as outlined above has been given considerable attention in the area of language modeling for applications like speech recognition, understanding and synthesis [14] [15] [16] where the task is to parse a string of speech units, phonemes or words, to obtain a sequence of words or sentences (called multigrams). However, segmentation techniques that are feasible there, including dynamic programming and Estimate-Maximize iterative procedures, are highly impractical in our case. This is because the segmentation problem posed here is larger by several order of magnitudes. Therefore, a different approach is adopted here. Instead of finding a strictly optimal solution, over all possible segmentations, the following is done. The segmentation algorithm attempts to maximize the *size* of each subframe subject to a limit on the size of the combined-symbol alphabet, hence, subject to a limit on complexity. This approach separates the segmentation process from the subsequent lossless coding step. It is simple, fast, readily realizable and effective in the sense that it produces a small number of large segments whenever possible. The subsequent Huffman coder is optimized for the segmentation output and, therefore, the coded bit stream is generally short.

The algorithm is based on mapping subframes to combined symbols C_m by using variable-radix arithmetic. Let $R_m > 0$ be a radix (basis) for the mth combined symbol. Then,

$$C_m = \sum_{n=0}^{N_m-1} S_{k_m+n}\ R^n{}_m \qquad (2)$$

Expression (2) is executed subject to:

$$R_m = \max_{n=0}^{N_m-2} S_{k_m+n} + 1 \qquad (3)$$

and

$$C_m < \mathbf{C} \qquad (4)$$

where $\mathbf{C}$ is the alphabet size of C_m. The constraint (3) makes the set $S_{k_m+n}, n = 0, .., N_m - 1$ uniquely decodeable from C_m (knowing N_m and R_m) while generating an integer C_m of a minimum decodeable value. The constraint (4) determines the size of the combined-symbol alphabet, thereby setting the size of the memory space required by the Huffman encoder. The algorithm increments the group size N_m until (4) is violated, at which point the $m-th$ symbol is fixed and the algorithm starts processing the next segment. The algorithm proceeds until the segmentation is complete, producing M combined symbols with $\sum_m N_m = N$. The number of subframes M varies from one main frame to another. Various ad-hoc strategies can be employed for parsing the frame. The simple ones simply scan the frame left-to-right or right-to-left. A new segment starts where the previous ends. More sophisticated algorithms can be based on identifying areas of activity in the current frame, parsing these areas first. This may be suitable for MDCT data where such areas are distinctly observable. We used the simple left-to-right method with contiguous parsing.

The data of the current frame is now represented by the triplet $\{ C_m, R_m, N_m \}$, the combined symbol, the radix and the segment size. Each member of this triplet should be encoded, losslessly compressed and transmitted to the receiver for perfect reconstruction of the original source. Since all three streams originate from the same source it is reasonable to assume that they are statistically redundant to a certain degree. Therefore it might be more efficient to encode the triplet as one unit or to try to reduce the number of streams by modifying the algorithm. The next section provides experimental data regarding the information rates of these symbols.

3. INFORMATION RATES OF S_N, C_M, R_M, N_M

For each symbol, two measures of average rates are used: the entropy and the rate of a trained Huffman coder. All rates are given in average number of bits per original source samples. An MDCT-transform coder was used to generate the stream S_n. The source sampling rate was 16 kHz for a full bandwidth of 7 kHz and frame size of 20 msec, implying a frame length of $N = 280$. The training material consisted of diverse audio passages including speech and music of various kinds. The total audio duration was about 17 minutes. As the coder was running, the segmentation algorithm generated the symbols C_m, R_m, N_m, as well as all the required statistics for computing the entropy and the Huffman rate for each symbol. The rate measurements were repeated for several values of $\mathbf{C}$, the alphabet size of the combined symbol. Figure 1 shows the overall average rate needed for transmission of all symbols as a function of $\mathbf{C}$. The Entropy rate reflects the minimum possible rate (assuming each stream is coded separately). The Huffman rate is that of a practical Huffman coder trained for each stream. As expected, the difference between the entropy and Huffman rates is large in the scalar case ($\mathbf{C} = 1$) and diminished in the vector case, as $\mathbf{C}$ increases. The figure shows that the overall coding rate of the VSVEC (including all streams) is about 75% of the scalar Huffman rate.

It is of interest to look at the segment sizes associated with the rates shown in figure 1. As shown in figure 2, the average segment size increases with $\mathbf{C}$. This explains the rate reduction observed in figure 1. As expected, the lossless coding is more efficient when applied to longer vectors of combined indices. Figure 3 shows the entropy and Huffman rates for the individual symbols as a function of $\mathbf{C}$. As the range increases, the rates of the segment size and the radix decrease and the rate of the combined indices increases.

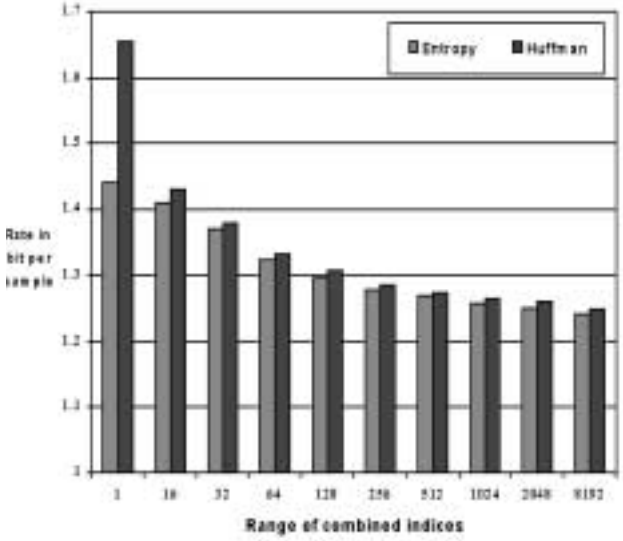

Figure 1. Total entropy and Huffman rates of all symbol streams as a function of the range **C**.

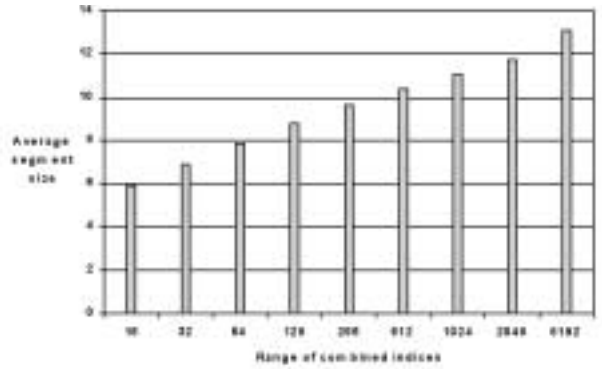

Figure 2. Average segment size N_{av} as a fuction of the combined symbol range **C**.

This is intuitively explained by noting that scalars N_m and R_m become less important when the combined index C_m represents longer vectors. Figure 3 also shows that the rates of N_m and R_m (the side information) constitute a significant part of the overall information, especially for lower index ranges. This motivates the discussion of the next section.

4. THE FIXED-RADIX APPROACH

Various approaches may be taken for reducing the rate of the symbol triplet defined above. Generally, it is possible to further combine two or more symbols into "super symbols" exploiting the their mutual redundancy. Alternatively, a simpler way is to make one (or more) symbol a constant. We have investigated several approaches. Limited by space, we will describe only one such method which uses a constant radix.

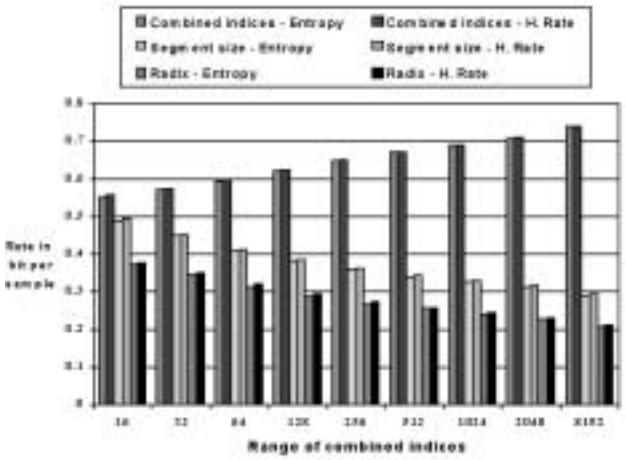

Figure 3. Entropy and Huffman rates of combined indices, segment sizes and radices.

Using a fixed radix R removes the need for its transmission. However, with a fixed radix, the segmentation usually generates shorter subframes which translates into a higher combined-index rate. The overall coding rate is hard to predict and can only be examined experimentally. With a fixed radix, the combined symbol is given by

$$C_m = \sum_{n=0}^{N_m-1} S_{k_m+n} R^n \tag{5}$$

Expression (5) is successively executed with increasing N_m, subject to:

$$R > \max_{n=0}^{N_m-2} S_{k_m+n} \tag{6}$$

and

$$C_m < \mathbf{C} \tag{7}$$

The segment size N_m is incremented until one or more of the above constraints is violated, at which point, a new segment starts. The search proceeds in a left-to-right direction. This algorithm guarantees a segment size of at least 1 and at most N. There are now only two streams to code: the combined symbols and the associated segment sizes. The performance of this scheme was measured versus the parameters R and **C** and is summarized in figure 4. Figure 4 shows a strong dependence on the value of the fixed radix R and (surprisingly) a weak and unstructured dependence on the symbol range **C**. In general, the rates are lower for higher radices. There is only a slight tendency of lower rates at higher ranges. It seems that some implicit relation between R and **C** affects the overall statics in favor of certain pairs (R, **C**). The best result of 1.089 bps is for the pair (25, 8192) followed by 1.091 bps for (25, 512), followed by 1.095 pbs for (20, 4096). These results are practically all the same. This indicates that, with this scheme, good performance can be obtained without the need for large memory (high complexity).

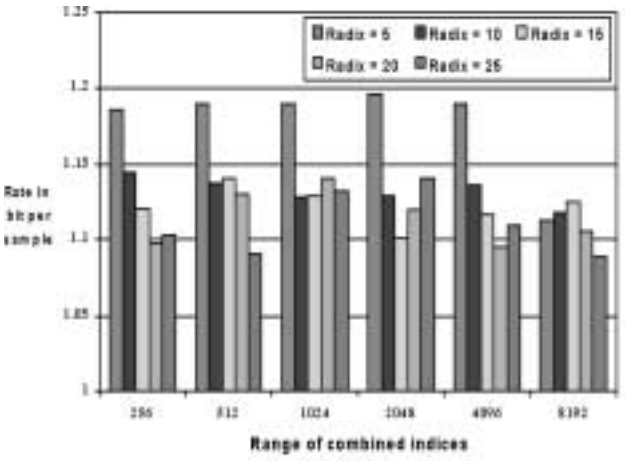

Figure 4. Total Huffman rates for the fixed-radix approach as a function of the combined symbols range $\mathbf{C}$.

Comparing these results to the data in figure 1 shows that eliminating the radix stream reduces the rates by 12% to 17% based on same range (same complexity) with the bigger rate savings obtained for the lower complexity cases. overall, the rate reduction is about 37% relative to scalar Huffman coding. Also, note that the fixed-radix procedure is less complex than the variable-radix one. This is because expressions (5) and (6) can be evaluated recursively using about as low as 7 arithmetic operations per source sample (the exact number depends on the implementation). Since the performance for various pairs $(R, \mathbf{C})$ is somewhat unpredictable, it may be useful to repeat the measurements on a denser grid in the $R, \mathbf{C}$ space to possibly find the best parameters for the intended application.

5. CONCLUSIONS

This paper discusses the concept of and an algorithm for variable-size vector entropy coding (VSVEC). The algorithm is based on complexity-constrained and size-unconst -rained fast segmentation and combination of a long stream of source symbols. The method is evaluated in the framework of an MDCT coder using arrays of 280 coefficients as an input frame of source symbols. It is shown experimentally that a coding rate reduction of about 37% over a simple scalar Huffman coder can be achieved. The segmentation algorithm requires about 7 arithmetic operations per sample which is well within the capability of today's DSP's, and CPU's. The required memory (ROM) should support Huffman tables of about 512 in size or about 2 kbytes.

Because VSVEC is based on coding high-dimensional vector-symbols, optimizing the Huffman coder over a short media passage results in a highly tuned coder and very high compression ratio. Therefore, VSVEC is an efficient compression method for storage applications.

Finally, the proposed VSVEC is not specific to MDCT coders. It is generic in nature and can be incorporated in various types of media coders.

REFERENCES

[1] D. Huffman, *A method for the construction of minimum redundancy codes*, Proc. IRE 40, pp. 1098-01191, 1952.

[2] G.V. Cormack, R.N. Horspool, *Algorithms for adaptive Huffman codes*, Inf. Proc. Letters, 18, 3 (Mar.), pp.159-165.

[3] J. Rissanen, G.G. Langdon, *Arithmetic coding*, IBM J. Res. 23, 2 (Mar.), pp. 149-162, 1979.

[4] A. Moffat, R.M. Neal, I.H. Witten, *Arithmetic coding revisited*, ACM Trans., On Information Systems, Vol. 16, No. 3, July 1998, pp. 256-294.

[5] S. A. Savari, R.G. Gallager, *Generalized Tunstall codes for sources with memory*, IEEE Trans. IT, Vol. 43 No. 2, pp. 658-667, March 1997.

[6] M.A. Gerzon, P.G. Craven, J.R. Stuart, M.J, Law, R.J. Wison, *The MLP lossless compression system*, AES 17th, Int.Conf. on High Qual. Audio Coding, pp. 1-15.

[7] P. Craven, M. Gerzon, *Lossless coding for audio discs*, J. audio Eng. Soc., Vol. 44 No. 9 pp. 706-720, Sep. 1996.

[8] P.A. Chou, T. Lookabaugh, R.M. Gray, *Entropy-constrained vector quantization*, IEEE Trans. Acoust, Sp. and Sig. Proc. 37(1), pp.31-42, Jan. 1989.

[9] J. Ziv, A. Lempel, *A universal algorithm for data data compression*, IEEE Trans., Inf. Theor., Vol. IT-23, pp.337-343, 1977.

[10] J.P. Princen, A.B. Bradley, *Subband/transform coding using filter bank designs based on time domain aliasing cancellation*, IACSSP'87, pp. 2161-2167, 1987.

[11] D. Sinha, J.D. Johnston, S. Dorward, S.R. Quackenbush, *The digital signal processing handbook*, Editors: V.K. Madisetti and D.B. Williams, CRC Press, 1998.

[12] S.R. Quackenbush, J.D. Johnston, *Noiseless coding of quantized spectral components in MPEG-2 advanced audio coding*, IEEE workshop on applications of signal processing to audio and acoustics, WASPAA'97, Session 3. Paper No. 3.

[13] PictureTel Corp. *Detailed description of the PTC (PictureTel Transform Coder)*, ITU-T Standardization sector, Study Group 15, Question. 6/15, Oct. 8-9, 1996.

[14] F. Jelinek, *Self-organized language modeling for speech recognition*, Reading in Speech Recognition, pp. 450-506, Morgan Kauffmann Pub. Inc., 1990.

[15] S. Deligne, F. Bimbot, *Language modeling by variable length sequences: theoretical formulation and evaluation of multigrams*, ICASSP'95 pp. 169-172, 1995.

[16] S. Roucus, M. Ostendorf, Herbert Gish, A, Derr, *Stochastic segment modeling using the Estimat-Maximize algorithm*, ICASSP'88 pp. 127-130, 1988.