CLASS-DEPENDENT AND DIFFERENTIAL HUFFMAN CODING OF COMPRESSED FEATURE PARAMETERS FOR DISTRIBUTED SPEECH RECOGNITION

Young Han Lee, Deok Su Kim, and Hong Kook Kim

Department of Information and Communications
Gwangju Institute of Science and Technology (GIST), Gwangju 500-712, Korea
{cpumaker, dskim867, hongkook}@gist.ac.kr

ABSTRACT

In this paper, we propose an entropy coding method for compressing quantized mel-frequency cepstral coefficients (MFCCs) used for distributed speech recognition (DSR). In the European Telecommunication Standards Institute (ETSI) extended DSR standard, MFCCs are compressed with additional parameters such as pitch and voicing class. The entropy of compressed MFCCs in each analysis frame varies according to the voicing class of the frame, thereby enabling the design of different Huffman trees for MFCCs according to voicing class, referred to here as class-dependent Huffman coding. In addition to the voicing class, the correlation in subvector-wise is utilized for Huffman coding, which is called subvector-wise Huffman coding. It is also explored that differential Huffman coding can further enhance a coding gain against class-dependent Huffman coding and subvector-wise Huffman coding. Based on the benefits above, hybrid types of Huffman coding by combining class-dependent and subvector-wise with differential Huffman coding are compared in this paper. Subsequent experiments show that the average bit-rate of subvector-wise differential Huffman coding is measured at 33.93 bits/frame, whereas that of a traditional Huffman coding which does not consider voicing class and encodes with a single Huffman coding tree for all the subvectors is at 42.22 bits/frame.

Index Terms— Distributed speech recognition, MFCC, Huffman coding, class-dependent, subvector-wise, differential coding

1. INTRODUCTION

As technologies associated with wireless network systems have advanced, the demand for wireless and mobile devices has also dramatically increased. These portable devices are typically small in size and difficult to manipulate. Thus, as a promising user interface to make them easier to use, speech recognition can take the place of a keyboard or a touch pad on these devices since voice input only requires a microphone [1]. A major problem, however, is that the high computational complexity of speech recognition is insufficient for most portable devices. Thus, a new approach, distributed speech recognition (DSR), was developed to implement speech recognition in portable devices. In particular, the European Telecommunication Standards Institute (ETSI) has published several versions of DSR front-end standards; the most recent version was the extended front-end defined in [2][3]. Basically, DSR splits the functions of speech recognition into a front-end and a back-end, where the former is performed in a portable device and the latter in a designated speech recognition server having a high computational power. The primary purpose of the DSR front-end is to extract speech recognition features such as the mel-frequency cepstral coefficients (MFCCs) that are commonly used for speech recognition. Then, the front-end compresses the MFCCs into as small a number of bits as possible, and then transmits them to the speech recognition server over a network.

When channel errors occur, however, DSR performance can degrade since the MFCCs decoded at the server become distorted; this problem can be somewhat overcome by adding a channel coder to the MFCC bitstream. In general, assigning more bits to a channel coder improves the quality of speech recognition [4]. Therefore, it is important to reduce the bits for MFCCs to accommodate such channel coding bits. Some techniques using Huffman coding have been proposed for quantizing MFCCs to reduce the overall bit-rate [5][6]. For example, the compression of MFCCs was done in [5] by applying a 2-dimensional discrete cosine transform (DCT) on blocks of feature vectors, followed by uniform scalar quantization, with subsequent run-length and Huffman coding. By performing such a series of compression schemes, the bit-rate for MFCC compression could be reduced to 15.6 bits/frame. Alternatively, the entropy coding proposed in [6] was applied to MFCCs extracted under the ETSI DSR framework, where one Huffman table [7] was used for the entropy coding of all MFCC indices. In this design, the Huffman coding reduced the MFCC compression bit-rate from 44 to 34.40 bits/frame.

In this paper, we first explore the entropy of compressed MFCCs and energy in order to further improve the efficiency of Huffman coding. In the extended DSR front-end, the feature parameters are composed of 13 MFCCs, a log energy, a pitch period, and a voicing class indicator for each frame. It is measured that the entropy of feature parameters varies according to the voicing class. This variance implies that MFCCs and log energy in the same class have higher redundancies and thus the bit-rates of the compressed MFCCs and log energy can be further reduced using an entropy coding method, such as Huffman coding. Therefore, we propose a class-dependent Huffman coding method to further obtain a coding gain over the compression of MFCCs and log energy according to voicing class. In addition to such class-dependent Huffman coding, it utilizes the property that the correlation between MFCC subvectors of the same frame is lower than the correlation of each subvector between frames, which brings us to consider Huffman coding in subvector-wise. After that, we also propose hybrid types of Huffman coding by combining

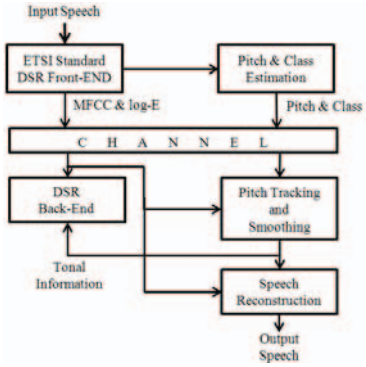

Figure 1: Block diagram of the ETSI DSR front-end defined in [2].

class-dependent and subvector-wise Huffman coding with differential Huffman coding based on the entropy comparison. Finally, we compare the bit-rate reduction for the different Huffman coding methods.

The remainder of this paper is organized as follows. In Section 2, the extended DSR front-end is briefly reviewed; in Section 3, we describe four different Huffman coding methods such as a traditional, class-dependent, subvector-wise, and differential Huffman coding. And then, we can combine class-dependent coding and subvector-wise coding with differential coding to take advantages of each method for compressing feature parameters. In Section 4, the performances of the proposed hybrid Huffman coding methods are measured and compared with those of the other Huffman coding methods. Finally, we conclude this paper in Section 5.

2. QUANTIZATION OF FEATURE PARAMETERS

In a typical DSR system, the front-end is located at the terminal and is connected over a data network to a remote back-end recognition server. Fig. 1 shows the extended DSR front-end standardized by ETSI [2]. The feature extraction is performed with 16 parameters, comprised of 13 MFCCs, a log-energy, a pitch period, and a voicing class indicator for each analysis frame. These extracted features are then compressed and transmitted to the server via a dedicated channel, where they are subsequently decoded and passed into the back-end for speech recognition.

The feature compression algorithm for DSR is as follows. The MFCCs and log energy are split into seven 2-dimsional subvectors and each subvector is directly quantized using a vector quantizer. Table 1 shows the feature pairing and the number of bits assigned to each subvector. The resulting set of index values after quantizing all the subvectors is then used to represent the corresponding speech parameters. The closest centroid of the vector quantizer can then be found using a weighted Euclidean distance to determine the index. In other words, $[y_i(m), y_{i+1}(m)]^T$, a subvector for the *m*-th analysis frame, is quantized as

$$d_j^{i,i+1}(m) = \begin{bmatrix} y_i(m) \\ y_{i+1}(m) \end{bmatrix} - q_j^{i,i+1} \quad (1)$$

and

$$idx^{i,i+1}(m) = \underset{0 \le j < N^{i,i+1}}{\operatorname{argmin}} \left\{ \left(d_j^{i,i+1}(m) \right)^T W^{i,i+1} d_j^{i,i+1}(m) \right\}, i = 0,2,\cdots,12 \quad (2)$$

Table 1. Split vector quantization codebook for feature pairing.

Codebook	Pairings	Size (bits)
$Q^{0,1} \sim Q^{10,11}$	$(C_1,C_2) \sim (C_{11},C_{12})$	6
$Q^{12,13}$	$(C_0,\text{log-E})$	8

where $q_j^{i,i+1}$ denotes the *j*-th centroid in the codebook $Q^{i,i+1}$, $N^{i,i+1}$ is the size of the codebook $Q^{i,i+1}$, $W^{i,i+1}$ is a weight matrix applied to the codebook search corresponding to the codebook $Q^{i,i+1}$, and $idx^{i,i+1}(m)$ denotes the codebook index chosen to represent the subvector $[y_i(m), y_{i+1}(m)]^T$. The size of each pairing is 64 for the subvectors of C_1–C_{12}, and 256 for the subvector of C_0 and log energy (log-E). Consequently, 44 bits are required to quantize all the feature parameters for each frame; these indices are then transmitted to the back-end.

In addition to speech recognition parameters such as MFCCs and log energy, pitch and voicing class information are used to indicate the voicing type of each frame, where each frame is classified into one of four classes such as non-speech, unvoiced speech, mixed-voiced speech, and (fully) voiced speech [3]. For non-speech or unvoiced speech, the pitch index is set to zero because there is no pitch period to be detected for these types of speech. In order to discriminate non-speech from unvoiced speech, the voicing class index is set to 0 or 1 for non-speech or for unvoiced speech, respectively. Conversely, the pitch index for mixed-voiced speech or (fully) voiced speech is set to an actual pitch period estimated from speech; the voicing class index is set to 0 or 1 for mixed-voiced speech or (fully) voiced speech, respectively.

3. PROPOSED HUFFMAN CODING METHODS

3.1. Traditional Huffman coding

A traditional Huffman coding applies the same Huffman table to each feature parameter without regarding to voicing class, which is similar to the work proposed in [6]. In fact, two Huffman tables are generated in this paper; one is for the MFCC subvectors and the other for the subvector of C_0 and log-E.

3.2. Class-dependent Huffman coding

The entropy of MFCCs and log energy varies according to the voicing class. This variance implies that MFCCs and log energy in the same class have higher redundancies and thus the bit-rate of the compressed MFCCs and log energy can be further reduced using Huffman coding that is designed differently according to voicing class. To this end, we classify the MFCC subvectors and the subvector of C_0 and log energy into four groups depending on their voicing class, and then construct two Huffman tables for each voicing class; one is for the MFCC subvectors and the other for the subvector of C_0 and log-E.

Fig. 2(a) shows a block diagram of class-dependent Huffman coding. First, the extracted MFCCs and log energy for a given analysis frame are quantized as described in Section 2. Then, MFCC subvector indices and the subvector index of C_0 and log-E are further compressed using the Huffman tables corresponding to the voicing class of the frame.

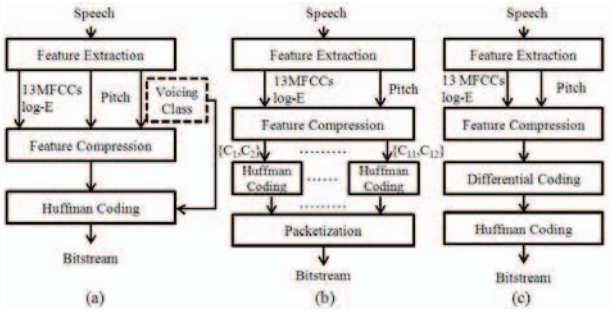

Figure 2: Block diagrams of (a) class-dependent Huffman coding, (b) subvector-wise Huffman coding, and (c) differential Huffman coding.

Table 2. Average entropy comparison (bits/frame) of each subvector for the different Huffman coding methods.

Method		C_1–C_{12}	(C_0,log-E)
Traditional Huffman coding		5.82	7.06
Class-dependent Huffman coding	Non-speech	5.37	3.49
	Unvoiced speech	5.63	6.49
	Mixed-voiced speech	5.66	6.46
	(Fully) Voiced speech	5.88	6.70
	Average	5.75	6.42
Subvector-wise Huffman coding	(C_1,C_2)	5.06	7.06
	(C_3,C_4)	5.31	
	(C_5,C_6)	5.61	
	(C_7,C_8)	5.62	
	(C_9,C_{10})	5.19	
	(C_{11},C_{12})	5.39	
	Average	5.45	7.06
Differential Huffman coding		4.89	5.10

3.3. Subvector-wise Huffman coding

The entropy of MFCCs and log energy varies according to the subvector. This variance implies that MFCCs and log energy also have higher redundancies compared to traditional Huffman coding and thus the bit-rate of the compressed MFCCs and log energy can be further reduced using Huffman trees that are designed differently according to subvector. Fig. 2(b) shows a block diagram of subvector-wise Huffman coding. The extracted MFCCs and log energy are quantized as described. Then, each subvector is further compressed using the Huffman tree corresponding to the subvector of the frame.

3.4. Differential Huffman coding

In order to reduce the bit-rate, we then investigate the redundancy of feature vectors in time that can be utilized. For this task, we first obtain the MFCCs and log energy for each frame, and then quantize them as described in Section 2. Next, we obtain the time difference of each compressed index, $\Delta idx^{i,i+1}(m)$, as

$$\Delta idx^{i,i+1}(m) = idx^{i,i+1}(m) - idx^{i,i+1}(m-1), \quad i = 0,2,\cdots,12 . \qquad (3)$$

Finally, the differential Huffman coding method shown in Fig. 2(c) is applied to $\Delta idx^{i,i+1}(m)$.

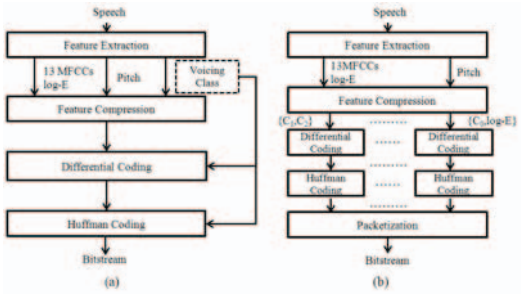

Figure 3: Block diagram of the hybrid Huffman coding methods according to (a) voicing class and (b) subvector-wise.

3.5 Entropy comparison

In order to investigate how much further the bit-rate of the compressed subvector indices can be reduced using Huffman coding, we evaluated the entropy of feature subvector indices according to traditional Huffman coding, class-dependent Huffman coding, and differential coding. Table 2 shows the measured entropy for each Huffman coding method. For this experiment, we used the TIMIT database [8] (clean, 16-bit, mono and sampled at 16 kHz) for training and testing, where 4,622 utterances and 1,680 utterances were used to generate the Huffman trees and to evaluate the performance, respectively.

As shown in the first row of the table, traditional Huffman coding required 5.82 bits/frame and 7.06 bits/frame for each of the MFCC subvectors, C_1–C_{12}, and the subvector of C_0 and log-E, respectively. On the other hand, 5.75 bits/frame and 6.42 bits/frame were required for each of C_1–C_{12} and the subvector of C_0 and log-E, respectively, in case of class-dependent Huffman coding. The classification percentages of non-speech, unvoiced speech, mixed-voiced speech, and fully voiced speech were measured at 4.06 %, 79.57 %, 0.81 %, and 15.56 %, respectively. These percentages were reflected to calculate the weighted average bits/frame of class-dependent Huffman coding. In case of subvector-wise Huffman coding, 5.45 bits/frame and 7.06 bits/frame were required for each of C_1–C_{12} and the subvector of C_0 and log-E. In addition, 4.89 bits/frame and 5.10 bits/frame were required for each of the subvectors of C_1–C_{12} and the subvector of C_0 and log-E when differential Huffman coding was applied.

3.6. Hybrid Huffman coding

From the entropy comparison in Table 2, it is shown that the MFCC subvectors have smaller entropy when differential Huffman coding is applied. It is expected from the table that hybrid types of Huffman coding by combining class-dependent and subvector-wise coding methods with the differential Huffman coding method. Figs. 3(a) and 3(b) show the proposed hybrid Huffman coding according to voicing class and subvector-wise, respectively. For the class-dependent differential Huffman coding, the differential coding is applied to the subvectors in frame-wise according to voicing class. On the other hand, the subvector-wise differential Huffman coding is applied in subvector-wise. Table 3 shows the entropy comparison of the hybrid types of Huffman coding. It is shown from the table that proposed hybrid Huffman coding according to subvector has smaller entropy than any Huffman coding method shown in Table 2 while the hybrid Huffman coding according to voicing class has a little larger entropy compared to differential Huffman coding.

Table 3. Average entropy comparison (bits/frame) of hybrid Huffman coding methods.

Subvector		Normal	Differential
Class-dependent differential coding	(C_1,C_{12})	5.78	4.95
	(C_0, log-E)	6.46	5.24
	Total	**40.90**	**34.94**
Subvector-wise differential coding	(C_1,C_2)	5.06	4.01
	(C_3,C_4)	5.31	4.12
	(C_5,C_6)	5.61	5.02
	(C_7,C_8)	5.62	5.22
	(C_9,C_{10})	5.19	4.74
	(C_{11},C_{12})	5.39	5.53
	(C_0, log-E)	7.06	5.10
	Total	**39.78**	**33.74**

Table 4. Average number of bits/frame of each subvector for the different Huffman coding methods.

Method	C_1–C_{12}	(C_0,log-E)
No entropy coding	6	8
Traditional Huffman coding	5.85	7.12
Class-dependent Huffman coding	5.78	6.46
Subvector-wise Huffman coding	5.49	7.12
Differential Huffman coding	4.91	5.14
Hybrid Huffman coding		
Class-dependent Huffman coding	5.00	5.26
Subvector-wise Huffman coding	**4.80**	**5.14**

4. PERFORMANCE EVALUATION

In this section, we compared the performance of the proposed hybrid Huffman coding methods with that of no entropy coding, traditional Huffman coding, class-dependent Huffman coding, subvector-wise Huffman coding, and differential Huffman coding in a view of their average number of bits per frame. For the original vector quantization, the average number of bits was measured at 44 bits/frame since 6 subvectors for C_1–C_{12} were quantized with 6 bits each and 1 subvector for C_0 and log-E was quantized with 8 bits.

Table 4 shows the comparison of average number of bits/frame required for the different Huffman coding methods. As shown in the last row of the table, the proposed subvector-wise differential Huffman coding method gave an average bit reduction of 1.20, 1.05, 0.95, 0.69, 0.11, and 0.20 bits/frame for the subvectors of C_1–C_{12}, compared to the average number of bits for no entropy coding, traditional Huffman coding, class-dependent Huffman coding, subvector-wise Huffman coding, differential Huffman coding, and class-dependent differential Huffman coding method, respectively. It was also shown from the table that applying subvector-wise differential Huffman coding method to the subvector of C_0 and log-E provided average reductions of 2.86, 1.98, 1.32, 1.98, 0.0, and 0.12 bits/frame, compared to the same respective conditions.

Finally, Fig. 4 summarizes the average number of bits/frame for the different Huffman coding methods including no entropy coding. As a result, there were 33.93 bits/frame when the proposed hybrid Huffman coding method was applied, which corresponded to bit-rate reductions of 10.07 bits/frame, 8.29 bits/frame, 7.20 bits/frame, 6.12 bits/frame and 0.67 bits/frame compared to no entropy coding, traditional Huffman coding, class-dependent Huffman coding, subvector-wise Huffman coding, differential Huffman coding, and hybrid Huffman coding according to voicing class, respectively.

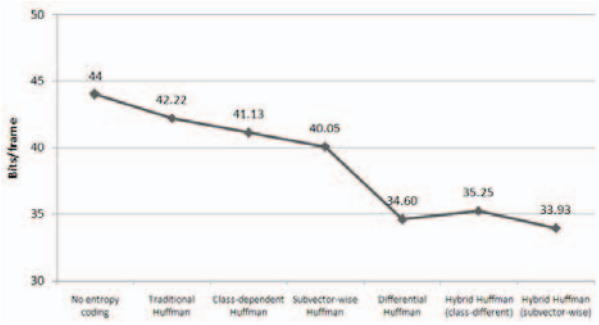

Figure 4: Performance comparison in the average number of bits for the different Huffman coding methods.

5. CONCLUSION

In this paper, we proposed a hybrid Huffman coding of MFCC feature vectors for DSR according to voicing class or subvectors and differential coding as a means of further reducing the bit-rate of compressed MFCCs. The proposed method was based on the subvector-wise property of feature distribution as well as the redundancy reduction in time differences of features. By combining these properties, we designed seven differential Huffman tables according to the subvectors. The proposed subvector-wise differential Huffman coding method provided an average bit-rate reduction of 10.07 bits/frame compared with the case when no entropy coding was employed in the DSR. Moreover, we had a bit-rate reduction of 0.67 bits/frame compared to differential Huffman coding.

6. ACKNOWLEDEGEMENT

This work was supported in part by the Korea Science and Engineering Foundation (KOSEF) grant funded by the Korea government (MEST) (R01-2008-000-10243-0), and by the Ministry of Knowledge Economy under the Information Technology Research Center support program supervised by the Institute of Information Technology Advancement (IITA-2008-C1090-0801-0017).

7. REFERENCES

[1] N. Srinivasamurthy, A. Ortega, and S. Narayanan, "Efficient scalable encoding for distributed speech recognition," *Speech Communication*, vol. 48, no. 8, pp. 888–902, Aug. 2006.

[2] ETSI ES 202 211, *Speech Processing, Transmission and Quality Aspects (STQ); Distributed Speech Recognition; Extended Front-end Feature Extraction Algorithm; Compression Algorithms; Back-end Speech Reconstruction Algorithm*, Nov. 2003.

[3] A. Sorin, *et al.*, "The ETSI extended distributed speech recognition (DSR) standards: Client side processing and tonal language recognition evaluation," in *Proc. of ICASSP*, pp. 129–132, May 2004.

[4] Z.-H. Tan, P. Dalsgaard, and B. Lindberg, "Automatic speech recognition over error-prone wireless networks," *Speech Communication*, vol. 47, nos. 1-2, pp. 220-242, Sept.-Oct. 2005.

[5] Q. Zhu and A. Alwan, "An efficient and scalable 2D DCT-based feature coding scheme for remote speech recognition," in *Proc. of ICASSP*, pp. 113–116, May 2001.

[6] B. J. Borgstrom and A. Alwan, "A packetization and variable bitrate interframe compression scheme for vector quantizer-based distributed speech recognition," in *Proc. of Interspeech*, pp. 578–581, Aug. 2007.

[7] D. A. Huffman, "A method for the construction of minimum-redundancy codes," *Proc. of the IRE*, vol. 40, pp. 1098–1101, Sept. 1952.

[8] J. S. Garofolo, *Getting Started with the DARPA TIM1T CD-ROM: An Acoustic Phonetic Continuous Speech Database*, National Institute of Standards and Technology (NIST), Gaithersburg, MD, Tech. Rep., 1988.