EFFICIENT SEGMENT QUANTIZATION OF LSP PARAMETERS FOR VERY LOW BIT SPEECH CODING

Minoru KOHATA¹, Ikuya MITUYA**, Motoyuki SUZUKI¹, and Shozo MAKINO**

*Chiba Institute of Technology, Narashino, 275-0016, Japan, kohata@net.it-chiba.ac.jp
**Tohoku University, Sendai, 980-8575, Japan

1. INTRODUCTION

This paper presents a new segment quantization (SQ) method for LPC coefficients, which is based on a new segmentation scheme. In order to design an efficient segment coder, a segmentation method using the self-similarity of LSP coefficients between several frames is employed. The segmentation is carried out by matching the input LSP frames with each segment in the codebook, and the codebook is trained at the same time with a method similar to the Lempel-Ziv coding method, which is one of the universal coding methods for discrete symbols. In the training process, the segment codebook is grown from null to the desired size. Two types of segment quantizer are designed based on the proposed segmentation method, and both methods can operate at low rates of below 7 bit/frame, with low complexity and low cepstral distortion of less than 2.4 dB.

2. LZ-LIKE SEGMENTATION

2.1. LZ method and its extension to SQ

The Lempel-Ziv coding method [3][4] is one of the universal coding algorithms that use a dictionary. In general, the encoding process for universal coding using a dictionary is as follows.

1: Parse the input sequence \( x(s,......) = \{x(s), x(s+1), ... x(0)\} \) which consists of finite discrete symbols, into the prefix \( x(s,t) \) with a relevant parsing rule.

2: Encode the prefix \( x(s,t) \) with the current dictionary.

3: Update the dictionary with \( x(s,t) \) according to some algorithm.

In decoding, the original sequence can be restored by retrieving the sub-sequence in the dictionary corresponding to the received code, if the dictionary is updated with the same algorithm as the coder. Lempel and Ziv proposed incremental parsing such that,

\[
x(t_{i-1}^j + 1, t_j) = x(t_{i-1}^j + 1, t_j) x(t_j) \quad \text{for some } j, \quad 0 \leq j \leq i - 1 \quad (1)
\]

\[
x(t_{i-1}^j + 1, t_j) \neq x(t_{i-1}^j + 1, t_j) \quad \text{for any } j, \quad 0 \leq j \leq i - 1 \quad (2)
\]

Equation (1) means that the sub-sequence (except for the last symbol) made by the parsing is matched with one of the past sub-sequences. And (2) means that all the sub-sequences made by the parsing are different from each other. The universal coding algorithms using this parsing method are referred to as Lempel-Ziv (LZ) algorithms. The LZ method has been shown to be asymptotically optimal for a stationary source [5].

Since the LZ method is a coding method for discrete symbols, it must be extended so that it can encode continuous parameters in order to quantize LSP coefficients. For this extension, an update method other than one needing incremental parsing is required for the dictionary (codebook). One such method is explained in section 2.2.

To quantize an LSP segment, two methods are described. One is a method in which a training LSP sequence is first coded by an LZ-like algorithm in the usual way, then SQ is done with an LSP segment codebook pre-trained in the above process. The other method operates like the LZ method; SQ is carried out with a codebook which is trained simultaneously. In the following, the details of both methods are described.
2.2. LZ-like SQ method 1

Figure 1 shows an algorithm of the proposed LZ-like method 1 (LZSQ1). In LZSQ1, the parsing algorithm of the original LZ method is used only for codebook training, and the usual SQ algorithm with MSE criterion carries out the SQ using the trained SQ codebook. As shown in Fig. 1, the LSP training sequence is fed into a buffer whose size is fixed, then matching between all the code segments in the codebook and all the segments in the buffer is done. Here, a parsing method other than an incremental parsing must be derived, because the input sequence does not consist of discrete symbols. Here a threshold value, TH, is introduced, and the segment which satisfies the condition, is parsed.

\[ x(1, t) = x_0(t), \quad t = \max \{ \arg(x_n(j)), 1 \leq j \leq L \} \]  

(3)

Where \( x_n(j) \) satisfies,

\[ \{ x_n(j) \} = \{ x(0, j) \} \quad \text{if} \quad \text{dist}(x(1, j), code(k)) < TH, 1 \leq k \leq M \]  

(4)

In Eq. (4), \( \text{dist}(\cdot) \) is the average distortion per frame, \( L \) is the maximum size of the input buffer in number of frames, and \( M \) is the number of the code segment currently obtained. As a measure of distortion, cepstral distortion is adopted here. This parsing method fetches the prefix part of the buffer, which satisfies the condition of Eq. (4), and whose length is the maximum. That is, the prefix segment in the buffer is recognized to be a match with a segment in the codebook. If the condition of Eq. (4) is not satisfied for any prefix segment in the buffer, the LZSQ1 method parses the prefix segment whose length is MINSEGLEN, which becomes the minimum length of the SQ code. Normally, MINSEGLEN is set to a small number.

The codebook training in LZSQ1 contains two processes; the first one is the modification of the existing code which matches the prefix segment. If code(k) matches the prefix segment \( x(1, t) \), then the code(k) is modified by Eq. (5).

\[ \text{code}(k) = \frac{\text{code}(k) \cdot \text{code}(k) + x(1, t)}{\text{code}(k) + 1} \]  

(5)

Where \( \text{code}(k) \) is the number of times \( \text{code}(k) \) is matched to the input segments. This is simply a calculation of a weighted average. The second process is to append a new code to the current codebook \( S \), by

\[ S = S + x(1, t + 1) \]  

(6)

If \( t+1 \) exceeds the input buffer length, this process is omitted.

In the LZSQ1 method, the codebook size is unlimited, but an extremely large codebook is not suitable for practical use. To avoid this problem, some methods were considered, and they are described in section 3.1.

2.3. LZ-like SQ method 2

The second LZ-like method is referred to as LZSQ2. In LZSQ2, the training of the codebook and the coding of the input LSP sequence are interlaced. The algorithm is almost the same as that for LZSQ1 except for two points. One is that LZSQ2 requires quantization of one LSP frame that is appended to the code segment matched with the input sequence, because a decoder requires the information on the segment code index and on the appended LSP frame to duplicate the codebook. Some efficient quantization methods have been proposed to date. From the viewpoint of bit rate reduction, MSVQ or Split-VQ is considered to be the most suitable for the quantization.

The second point that is different from LZSQ1 is that the LSP frame quantized in the above-mentioned process must be used to update the codebook at the coder. This makes the two codebooks, the codebook of the coder and that of the decoder, the same throughout the coding process.

As in LZSQ1, the codebook size must be limited somehow. Furthermore, the total bit rate may be reduced if the bits for the quantization of the additional LSP frame can be saved. Some expedients for these operations are described in section 3.2.

3. EXPERIMENTS

3.1. Experimental Results for LZSQ1

The LZSQ1 method makes an SQ codebook by the algorithm mentioned in the last section, but for the codebook training, some external parameters must be given, such as TH,
MINSEGLEN, and the maximum codebook size. The best values of TH and MINSEGLEN must be found through experiment. As for the maximum codebook size, the LZSQ1 method inherently makes an SQ codebook whose size is unlimited, but some limitation is required for practical use. Therefore, the following structure of code segment is adopted,

\[ s = \{LEN, COUNT, LSP[], SEQ\} \quad (7) \]

where LEN is the segment length, COUNT is the frequency that the code segment is matched with input segment(s), LSP[] is the body of the segment data, and SEQ is the sequential number which is incremented by one with each new code segment generation. The limitation of the codebook size is carried out by the following rule.

1. If the number of the code exceeds the limit, the segments whose COUNT is minimum are searched.
2. Among the segments found in the above search, the segment which has the minimum value of SEQ is purged.
3. A new segment is appended to the codebook whose COUNT is set to one, and whose SEQ is set to the "newest."

Using this rule, an old and infrequently used code segment is purged and overwritten by a new code segment, and thus the size of the codebook is efficiently limited.

Figure 2 shows experimental results for the LZSQ1 method. The experimental conditions are described in Table 1. From these results, LSP coefficients can be segment quantized at a 600 bit/s (6.0 bit/frame), with cepstral distortion of less than 2.2 dB. For reference, Split-VQ (4:6) and MSVQ (2 stages) are applied to the same test set and the result are shown in Figure 2. LZSQ1 can quantize the test set at the same bit rate with lower cepstral distortion compared with Split-VQ and MSVQ in all cases.

Figure 3 shows the number of times a code segment is matched to input segments, classified by segment length. The code segment whose length is two is most frequently used (MINSEGLEN=2). The bit rate for the code segment can likely be reduced if some entropy coding (such as Huffman coding) is used, although none was used in this experiment.

### 3.2. Experimental Result for LZSQ2

The LZSQ2 method does not require an external segment codebook, rather, the training interlaced with the parsing makes the codebook internally. As in the LZSQ1 method, LZSQ2 requires external parameters for its operation. Since the LZSQ2 method requires quantization of one LSP frame appended to the code segment, an efficient quantizer must be used. From the viewpoint of bit rate reduction, vector quantization is preferable, but even a Split-VQ or an MSVQ takes about 20 bit/frame [6], and this increases the total bit rate. To solve this problem, a modified MSVQ is introduced. Figure 4 shows the modified MSVQ quantizer. In this vector quantizer, the difference between the last LSP frame of the code segment and the LSP frame to be appended to the code segment is vector quantized as the second stage VQ. This method requires only half the bits of ordinary MSVQ and has low quantization distortion.

Figure 5 shows experimental results for the LZSQ2 method. The experimental conditions are described in Table 2. As test sets, an LSP set made of a single speaker's speech, and one made of multiple speakers' speech were used. From these results, it is apparent that the LZSQ2 method can operate at 700 bit/s (7.0 bit/frame), with cepstral distortion of less than 2.4 dB. As in Figure 2, Split-VQ and MSVQ are applied to the same test set.
and the results are shown. LZSQ2 can quantize the test set at the same bit rate with lower cepstral distortion compared with Split-VQ or MSVQ in all cases. Performance is better for the single speaker compared with that for the multiple speakers because the codebook is adapted to a single speaker. The performance of LZSQ2 does not exceed that of LZSQ1, but it requires no statistical information from the input source. Thus, its merit is similar to that of Lempel-Ziv coding as a universal coding.

Figure 6 shows the number of times the code segment is matched to input segments in the same manner as Fig. 3. For the same reason, the bit rate can be reduced if entropy coding is used, although it was not used here.

![Figure 4: Vector quantization with modified MSVQ.](image)

![Figure 5: Cepstral distortion and bit rate by LZSQ2.](image)

![Figure 6: Number of times a code segment is matched in LZSQ2.](image)

4. CONCLUSION

We have presented new segment coders for LSP quantization in very low bit rates. By introducing an LZ-like parsing method, which utilizes the self-similarity of LSP frames, an efficient segmentation can be realized. We proposed two segment quantizers, LZSQ1 and LZSQ2, and both were able to operate at a bit rate lower than 7bit/frame with cepstral distortion of less than 2.4 dB. Although LZSQ2, was not able to outperform LZSQ1, it requires no pre-trained segment codebook. This advantage is inherited from Lempel-Ziv coding as a universal coding, and it is expected that LZSQ2 can be applied to coding parameters other than LSP coefficients.

5. REFERENCES


