On the Efficient Compression Algorithm of the Voice Traffic

*Gyoun-Yon Cho and *Dong-Ho Cho

Abstract

The LZW algorithm has some redundancies for voice traffic. Thus, the V-LZW(Voice LZW) algorithm that decreases the redundancies of the LZW algorithm is suggested for efficient compression of voice. The V-LZW algorithm uses the differential method that reduces the codeword size as well as the length of repeated string. According to the simulation results, it could be seen that the performance of V-LZW algorithm is better by 35% to 44% in voice compression ratio than that of the conventional modified LZW algorithms. Compared with conventional 32 kbps ADPCM coding with 26 dB SNR, V-LZW has 21 kbps transmission rate, no additional quantization error and simple hardware complexity.

Ⅰ. Introduction

Recently, due to the progress and increasing variety of telecommunication service, the amount of data transferred through the communication channel is being increased rapidly. Therefore, the issues on the efficient transmission mechanism of various media stream is actively investigated. While there are many studies on the compression of data traffic, the compression of voice stream is not much studied unlike the speech coding such as waveform coder and vocoder. So, in this paper, the simple common compression algorithm which could be used to maximize the efficiency of the voice multiplexer is studied.

Many techniques about data compression are being studied nowaday. LZW compression technique based on universal compression technique is recommended as ITU-T V.42 bis compression algorithm, because of its good efficiency[1]. However, this LZW compression technique has some redundancies. That is, the conventional data compression algorithms have not taken into account voice traffic. When the voice traffic is sampled based on 64 kbps PCM encoder, its repetition cycle becomes more than 80 samples[2]. Thus, the conventional data compression algorithm which has 20 characters in string is inefficient for voice traffic which has long repetition cycle. Therefore, in this paper, the differential method is applied to the PCM coded voice traffic. Then, the compression is done based on LZW compression algorithm.

In this paper, in order to achieve efficient voice compression ratio in the voice integrated multiplexer, the new compression algorithm which could compensate for these inefficiencies of the conventional compression algorithm is proposed and analyzed through computer simulation. Following this introduction, in section 2, the LZW compression algorithm and the conventional modified compression algorithms are surveyed. Also, in section 3, the new compression algorithm which reduces the redundancies of the conventional compression algorithm is proposed. Moreover, in section 4, the newly proposed compression algorithm is compared with the conventional compression algorithms through the computer simulation. Finally, conclusions are made in section 5.

Ⅱ. The LZW Algorithms and Problems

1. LZW algorithm

LZW(Lempel Ziv Welch) is the algorithm that uses the conversion table mapping incoming string into the codeword. In LZW codeword table, the codeword has the prefix of string. That is, when there is a string wK composed of a string w and a character K, K and w are in the table. Here, K is called as extend character and w is called as prefix string[1], [3], [4], [5], [6], [7].

LZW codeword table includes the strings compressed previously. It means that the strings of codeword table have been trained already, and that the codeword table reflects the statistical property of the data stream previously applied. The encoding procedure of LZW algorithm is as follows.

1.1 The encoding procedure of LZW algorithm

The encoding procedure of LZW algorithm is as follows.

*Department of Computer Engineering, Kyung Hee University
Manuscript Received: March 19, 1997.

1) The initializing step of codeword table:
Register 256 characters as the initial codewords of codeword table. That is, initialize each code by ASCII character values whose ranges are from 0 to 255. Also, initialize the codeword variable cl to 256.
2) Read the first input character from the input data and set it to the prefix (w).
3) Read the next input character, and set it to extend character (K).
i) When no input exists in the next sequence, find the codeword matched to w and encode it to the binary code of $\lceil \log(cl+1) \rceil$ bits. (Here, $\lceil \log(cl+1) \rceil$ means a minimum integer that is bigger than $\log(cl+1)$.)
4) When wK exists in the codeword table, replace wK with a new prefix (w) and go to step 3).
5) When wK does not exist in the codeword table, find the codeword matched to w and encode it to the binary code of $\lceil \log(cl+1) \rceil$ bits, then go to following step.
i) Assign cl that is the codeword of wK to the codeword table, and register as a new string.
ii) Increase cl by 1 for the next codeword assignment.
iii) Replace K with a new prefix w, and go to the step 3).

1.2 The decoding procedure of LZW algorithm

The decoding procedure usually follows the reverse order of the encoding procedure. While the decoding procedure is activated, it has the same codeword table as the encoding procedure. Also, when encoded codewords are released, each codeword is decoded into a prefix string and an extend character by the contents of codeword table. Then, the extend character is outputed to stack and the prefix is decoded again to a prefix and an extend character. Here, when an extend character is separated from a prefix and outputed, it uses stack to change the order of output characters since the decoding order of extend characters is the reverse order of the encoding.

2. VV(Variable to Variable)-LZW algorithm

As mentioned above, in LZW algorithm, all codewords of strings generated during the encoding procedure are encoded to a binary code of $\lceil \log(cl+1) \rceil$ bits. That is, when the value of cl is from 256 to 511, the value of codeword is encoded into 9 bits. Also, when the value of cl is from 512 to 1023, the value of codeword is encoded into 10 bits. Thus, this algorithm has much redundancies in output bit stream.

To remove this redundancy of an output bit stream, the variable length LZW algorithm is used. In VV-LZW, a limit value is chosen among the values from codeword 0 to cl-1. For the codeword value between 0 and limit value, it is encoded to a binary code of $(\lceil \log cl \rceil - 1)$ bits. Also, for the codeword value between limit value and cl-1, it is encoded to a binary code of $\lceil \log(cl+1) \rceil$ bits.

3. The problem of voice compression

LZW encoding algorithm takes into account only English text. So, while it is very efficient in data compression, it is inefficient in voice compression. Thus, to compensate for this inefficiency, the new voice compression algorithm is required.

Ⅲ. The New Encoding Algorithm

From the analysis result for the PCM sampled voice samples, it could be seen that 99% of the differences between adjacent sample values is less than 32. Therefore, the number of codeword required is only 64. So, it is possible to express a codeword based on 6 bits.

1. Initializing Step

In the initialization step, register the codeword of 64 codes and then initialize cl with 65 to indicate the next character that the codeword to be registered should be inserted into the 65th row of the codeword table. Fig. 1 shows the configuration of the codeword table for the voice respectively.

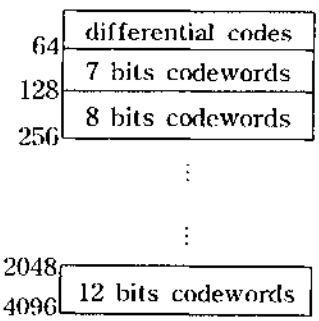

Figure 1. The configuration of the codeword table for the voice

2. Encoding Step

2.1 Voice encoding

In encoding of the voice, to apply the differential method, the following procedures are performed.

1) After subtracting the current value that is the value of the current sample from the old value that is the value of the previous sample, save the remainder as

temporary value.

2) Investigate the size of temporary value.

i) $0 \leq$ temporary value $< +31$,

Proceed to the step 3).

ii) $-32 <$ temporary value < 0,

After taking 2's complement of temporary value expressed as 6 bits, proceed to the step 3).

iii) $-32 \geq$ temporary value > -160,

Output 32 to indicate that temporary value is smaller than -32. Then, after dividing temporary value by 2 and calculating its absolute value, proceed to the step 3).

iv) $+31 \leq$ temporary value < 159,

Output 31 to indicate that temporary value is bigger than $+31$. Then, after dividing temporary value by 2, proceed to the step 3).

3) Return the value of temporary value.

2.2 The output step of encoded codeword

In order to reduce the size of encoded codewords, the following procedures are performed in the output step of encoded codewords. When encoded codewords are released, the following formula must be confirmed in order to request the right string. Here, if the number of codewords currently assigned is c1 and the limit value that can reduce the size of bits is limit, following equation is derived.

$$(c1 + limit) \leq 2^{\lceil \log(c1+1) \rceil} - 1 \quad (1)$$

Here, if $c3 = 2^{\lceil \log(c1+1) \rceil}$, equation (1) becomes as follows.

$$limit = c3 - c1 - 1 \quad (2)$$

Thus, when codewords are changed into bit stream based on the compression algorithm and the codeword is smaller than limit value, output the codeword expressed with $\lceil \log(c1+1) \rceil$ bits. Otherwise, output the codeword expressed $\lceil \log(c1+1) \rceil$ bits.

For example, when c1 is 750, the value of limits is calculated by using equation (2).

That is, $limit = 1024 - 750 - 1 = 273$. Therefore, if a codeword is between 0 and 273, encode and output it based on 9 bits. However, if a codeword is between 274 and 749 during encoding, add 274 to the codeword. Then, encode and ouput based on 10 bits.

During decoding, the codeword bits are read in 9 bits. Then, if this value is smaller than 274, the value is taken as a codeword. Also, if it is bigger than 274, the codeword bits are read in 10 bits and the subtraction of 274 from this value is done. When c1 is 750, the example of output bit stream is shown in Table 1.

Table 1. The example of output codeword when c1 is 750

encode codweord	output codeword bits	decimal value
0	0 0 0 0 0 0 0 0 0	0
1	0 0 0 0 0 0 0 0 1	1
2	0 0 0 0 0 0 0 1 0	2
.	.	.
.	.	.
273	1 0 0 0 1 0 0 0 1	273
274	1 0 0 0 1 0 0 1 0 0	548(274+274)
275	1 0 0 0 1 0 0 1 1 0	549(274+275)
.	.	.
.	.	.
749	1 1 1 1 1 1 1 1 1 1	1023(274+749)

Ⅳ. Simulation and Discussions

For the performance evaluation of the new compression algorithm proposed in this paper, V-LZW algorithm is implemented using C language in 486 PC. Also, in order to compare it with conventional algorithms, LZW and VV(Variable-to-Variable)-LZW are implemented by C language in 486 PC. For the voice traffic, real conversational digitized speech data based on 64kbps PCM encoder is considered.

The differential method is used to increase the compression efficiency. The differential method converts the PCM encoded voice sample into the differential code. Fig. 2 and 3 show the probability distributions before and after the conversion. It could be seen from Fig. 2 and 3

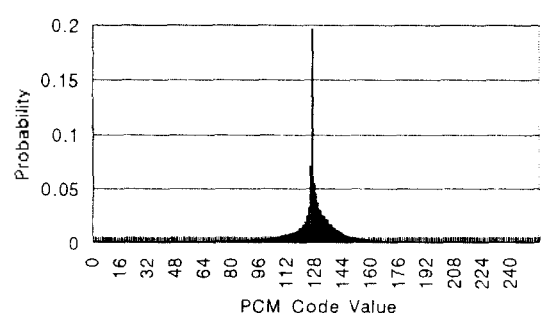

Figure 2. The probability of each PCM code before the differential conversion

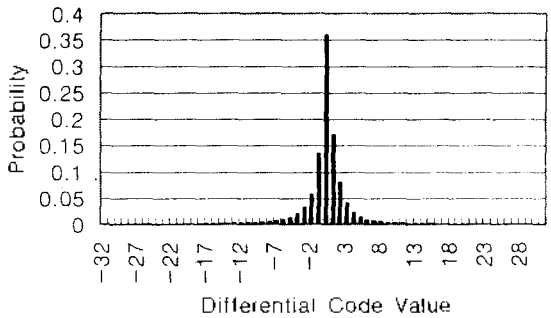

Figure 3. The probability of differential code after the differential operation

that the maximum probability of the difference after the conversion is about 0.36, while the maximum probability of the PCM code before the conversion is about 0.20. From the above results, it could be seen that the convergence rate of data is increased after conversion. Therefore, the code repetition rate could be increased by using the differential conversion.

Table 2. The compression ratio of voice

	sample 1	sample 2	sample 3	sample 4	average
LZW	2.15	2.62	2.43	1.71	2.23
VV-LZW	2.22	2.71	2.51	1.78	2.31
V-LZW	3.00	3.66	3.35	2.5	3.13

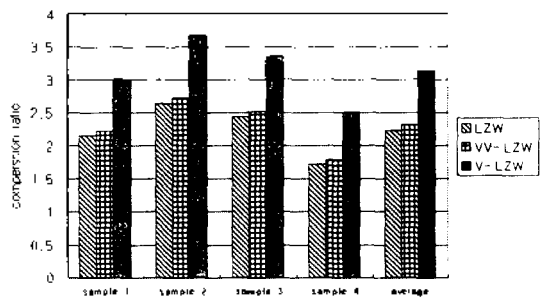

Figure 4. The compression ratio diagram of voice

Each digitized speech is sampled in real conversational environments for 30 minutes. Table 2 and Fig. 4 show the compression ratio of each algorithm. It could be seen that the performance of V-LZW algorithm is superior by 35% to 44% to that of the conventional algorithm. Also, it could be seen from Fig. 4 that the compression ratio is high in the sample 2 and low in the sample 4. This phenomenon could be explained by the fact that sample 2 has smaller difference between adjacent samples than sample 4.

Fig. 5 shows the probability distributions of the differential code for each sample. In this figure, it could be seen that the sample 2 has the highest speech correlation and the sample 4 has the lowest speech correlation. From Fig. 4 and Fig. 5, we can see that high speech correlation generates high compression ratio.

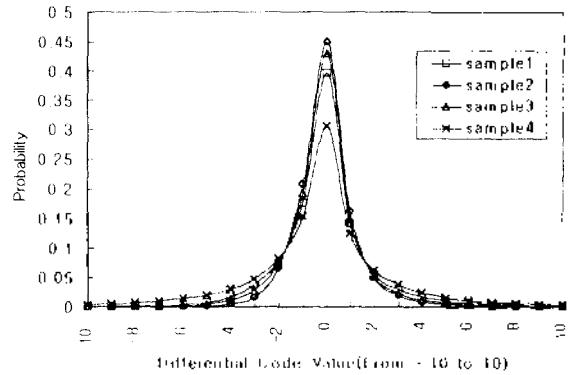

Figure 5. The probability distribution of differential code for each samples

Table 3 represents the number of changed samples when the compressed voice traffic based on the V-LZW algorithm is decoded into the PCM code. It could be seen from this table that the average ratio of the changed samples is 0.026%. Such a minor loss of voice sample due to the V-LZW compression algorithm is ignorable to the human ears.

Table 3. The number of samples changed due to the compression algorithm

	file size(samples)	changed sample	changed %
sample 1	65000	4	0.006
sample 2	260000	49	0.019
sample 3	455000	7	0.002
sample 4	650000	308	0.047
total	1430000	368	0.026

The compression ratios of both voice traffic with talkspurts only and voice traffic with talkspurts/silence portion are shown in Fig 6. In this figure, that the sample 1 has little silence period, and the sample 2 includes 40% silence period. Then, the compression ratio of the V-LZW algorithm is increased as silence period is larger since code repetition ratio is increased as the difference between adjacent samples is reduced.

Fig. 7 shows the result in the case that the input data which is converted into ADPCM are compressed by V-LZW algorithm. Compared with the previous result,

which shows the result in the case that the input data is converted into the differential value and compressed by V-LZW algorithm, the compression ratio of ADPCM code has less variance than the compression ratio of differential value. That is, the range of the compression ratio of differential value for PCM is from 2.5 to 3.66. On the other hand, the range of compression ratio of ADPCM code for PCM is from 2.97 to 3.24. This is due to the fact that differential value has more dynamic property than ADPCM code.

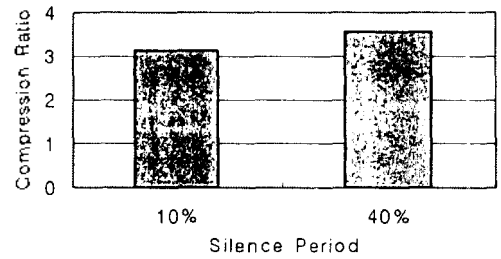

Figure 6. The compression ratio of voice traffic

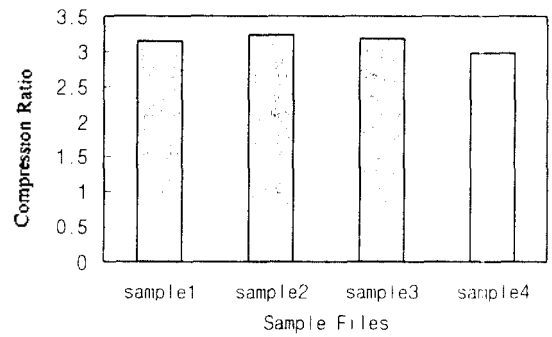

Figure 7. The compression ratio of ADPCM code

V. Conclusions

The LZW algorithm has several redundancies in applying it to voice. In order to reduce the redundancies of data traffic, various conventional modified LZW algorithms are suggested. However, these algorithms could not reduce the redundancies related to the voice traffic, because it does not take into account the characteristic of voice traffic. Thus, in this paper, V-LZW algorithm that could reduce the redundancies of voice traffic in voice multiplexer is suggested.

V-LZW algorithm uses the differential conversion method to minimize the codeword size as well as the length of repeated string. According to the simulation results, it could be seen that the performance of V-LZW algorithm is superior to that of conventional compression algorithms by 35% to 44% in voice compression ratio. Also, V-LZW algorithm has 21 kbps transmission rate, no additional quantization error and simple hardware, compared with ITU-T 32 kbps ADPCM with 26dB SNR.

In addition, in this paper, we have considered the compression of ADPCM code as well as the compression of differential value. The compression ratio of ADPCM code for PCM is almost the same as the compression ratio of differential value for PCM.

References

1. ITU-T Recommendation V.42 bis, "Data Compression Procedures for Data Circuit Termination Equipment using Error Correction Procedures", ITU-T, 1992.
2. ITU-T Recommendation G.711, "Pulse Code Modulation (PCM) of Voice Frequencies", ITU-T, 1988.
3. Timothy Bell, "Modeling for Text Compression", *ACM Computing Serveys*, Vol.21, No.4, pp. 557-591, December 1989.
4. Terry A. Welch, "A Technique for High-Performance Data Compression", *IEEE Computer*, 17.6, pp. 8-19. 1984.
5. Edward R. Fiala, "Data Compression with Finite Windows", *Computer of ACM*, Vol.32, No.4, pp. 490-505, April 1989.
6. Held, *Data Compression*, WILEY, 1993.
7. Jacob Ziv, Abraham Lempel, "Compression of Individual Sequences via Variable-Rate Coding", *IEEE Tran. on Information Theory*, Vol.IT-24, No.5, pp. 530-536, September 1978.

▲Gyoun Yon Cho

Gyoun Yon Cho was born in Chungnam, Korea, on Feb, 7, 1968. He received the B.E. and M.E. degrees in computer engineering in 1993 and 1995 respectively from Kyung Hee University, Korea, and is studying toward Ph.D. degree at the same university. His special interests include communication protocol engineering and implementation, personal communication service, ATM, wireless LAN, voice/data communication and FPLMTS etc.

▲Dong Ho Cho

Dong Ho Cho was born in Seoul, Korea, on April 3, 1956. He received the B.E. degree from Seoul National University, Seoul, Korea, in 1979 and the M.E. and Ph.D. degrees from Korea Advanced Institute of Science and Technology, Seoul, Korea, in 1981 and 1985, respectively. He joined the Communication Research Laboratories, KAIST, in 1985 and was engaged in the research and development of speech processing, packet switched data network and ISDN. From 1987, he has been a professor at the Department of Computer Engineering, Kyung Hee University, Korea. His research interests include mobile communication systems, computer security, communication network and multimedia systems. He is a member of the IEEE and IEICE.