Research on Lossless Data Compression Technology

Fuwei Lv, Xin Zhong

College of Mechanical and Electronic Engineering, Shandong University of Science and Technology, Qingdao, 266590, China

Abstract

The implementation of data compression technology can be divided into software compression and embedded hardware compression. The software compression operation is large, requires large data buffer space, and does not have real-time performance. This does not meet the requirements of real-time transmission of telemetry system; embedded hardware compression overcomes the above shortcomings of software compression and is widely used in telemetry systems. Data compression is divided into lossy compression and lossless compression. The high reliability and data accuracy of the aerospace telemetry system determine that only lossless compression can be used to facilitate the analysis of data by technicians and make correct judgments.

Keywords

Software, real-time, lossless compression.

1. Introduction

The development of the aerospace industry has put forward higher requirements for telemetry systems. The increase of electronic equipment and the improvement of the accuracy of sensor parameters have intensified the growth of information data. At present, the telemetry system widely equipped in China adopts S-band and PCM coding, and the single-link bandwidth is only 2Mbps. It is necessary to improve bandwidth utilization and alleviate bandwidth shortages. For a large number of sensors arranged in a spacecraft, the signals generated can be divided into slowly varying data (0 Hz to 10 Hz) and fast variable data (>10 Hz) according to the frequency range, wherein the speed change data accounts for 85% of the total data volume. Above [1]. The purpose of data compression is to transmit data more efficiently, reduce channel occupancy, and transmit more information in existing fixed rate communication systems. The implementation of data compression technology can be divided into software compression and embedded hardware compression. The software compression operation is large, requires large data buffer space, and does not have real-time performance. This does not meet the requirements of real-time transmission of telemetry system; embedded hardware compression overcomes the above shortcomings of software compression and is widely used in telemetry systems [2]. Data compression is divided into lossy compression and lossless compression. The high reliability and data accuracy of the aerospace telemetry system determine that only lossless compression can be used to facilitate the analysis of data by technicians and make correct judgments [3]. Here, several lossless data compression techniques, such as Huffman coding, Rice coding, and LZW coding, are proposed for the related problems of lossless data compression. By analyzing these coding methods in detail, the advantages and disadvantages of the compression algorithm are compared and analyzed. Finally, a lossless compression technique that can improve the communication capability of the bandwidth-limited channel and the most time-efficient transmission of remote sensing data is obtained.

Keywords: lossless compression Huffman coding Rice coding LZW coding real-time

2. Rice compression algorithm

2.1 Basic working principle of Rice compression algorithm [4]

R ice lossless data compression technology is a compression standard recommended by the International Spatial Data System Advisory Committee (CCSDS) for use in AOS systems. It has its

own unique features in terms of algorithm, not only has a high compression ratio, but also has a delay. Small, fast, and highly error-resistant, the main part includes pre-processor and adaptive entropy encoder.

2.2 Preprocessor

The function of the preprocessor is to convert the original input data into data that can be more beneficial to entropy coding, and the decorrelation performance of the preprocessor is one of the important steps to determine the coding efficiency. For lossless compression, the pre-processing process must be reversible. The general preprocessor consists of a predictor and a predictive error converter. The best predictor is selected to complete the prediction, and then each of the prediction errors can be transformed by the converter into a non-negative integer suitable for entropy encoder processing.

2.3 Adaptive encoder

The CCSDS entropy coder is a set of variable length coder. Finally, for each block unit, one of the most efficient encoders is selected by adaptive, and then the encoded result is transmitted together with the identifier of the encoder. Entropy coding of block units. Because each block can adaptively select the most effective entropy coding method, the RICE algorithm introduced by CCSDS can fully adapt to the variation of the statistical characteristics of the remote sensing image source and improve the coding efficiency. The coding modes of CCSDS entropy coder include the following four types: basic sequence coding, sample decomposition coding, low entropy coding and uncompressed coding. Like the Huffman algorithm, Rice algorithm also uses variable length symbol codewords as the basic compression method. . However, the simpler regularity of the FS code makes it unnecessary to perform the usual lookup process during encoding and decoding operations. Although this encoding process is simple, if the data symbols are large, the FS encoding will inevitably be too long, and the compression ratio will be lowered. Therefore, in the Rice lossless data compression algorithm, the concept of the preprocessor is first introduced, so that the source data is preprocessed before encoding. The main idea of preprocessing is to de-correlate the data and then transform the result into The process of positive integers with better probability distribution, the pre-processed data, the general values are very small, and it is convenient to adopt FS coding. For the data with high depreciation, the sample decomposition and classification process are performed. The Rice algorithm also cites mechanisms such as zero-value block processing, quadratic expansion processing, and uncompressed processing, which greatly improves compression efficiency and real-time performance.

2.4 Design an effective preprocessor for space applications

As a compression processor for spaceborne space applications, the effectiveness of the method should be considered in the algorithm design. At the same time, the reliability of engineering application must be fully considered, and the prevention of error diffusion is the primary problem to be solved in the reliability design of the algorithm. Therefore, in the design of the algorithm, a certain range of images must be separated, and the truncation of the prediction correlation is performed to limit the bit error rate to a certain image range. Predictive technology is a relatively effective method in compression processing. It usually has techniques such as texture prediction method, unit delay method and bump prediction method. For the above prediction model, if the error occurs during the decompression process, a, b, c, d If any pixel has an error, it may directly cause an x error, and then pass the error to x, and if the cut-off predictive continuity is not considered during the compression process, the error will continue to be propagated, resulting in The quality of the entire image was destroyed. In the scheme design, the error-resistance capability can be improved by periodically cutting off the context correlation of the prediction processor, and the sample interval is set within an appropriate range.

2.5 Preprocessor based on unit delay method

Only the Unit-Delay method in prediction techniques is introduced in the CCSDS Recommendation, in which the expected value $x\wedge i = xi-1$.

The function of the preprocessor is to convert the original input data into data that can be more efficiently entropy encoded, and the decorrelation performance of the preprocessor is one of the important steps to determine the coding efficiency. To ensure compression and loss, the preprocessing must be reversible. . The preprocessor is set here to consist of a predictor and a predictive error converter. A more sophisticated approach can provide very efficient compression performance, but at the expense of complexity. The preprocessor based on the unit delay method is a linear first-order unit delay predictor whose output Δi is the difference between the input data and the previous data. Each unit of data in the source data (such as a character in a message) is called a sample, and each sample is n bits. The source data is first divided into sample blocks of length J before inputting the preprocessor to improve the adaptability of the encoder to changes in entropy. The CCSDS Recommendation specifies J = 8 or 16, and $8 \leq n \leq 32$. The so-called prediction is to find the difference between the current data value xi minus the expected value x∧xi, that is, the prediction error. On the surface, the difference between the two n-bit numbers will produce a value of n+1 bits with a dynamic range between [-2n+1, 2n-1]. The purpose of error mapping is to transform each prediction error into a non-negative integer suitable for entropy coder processing. According to Shannon's theory of informatics, the greater the probability of occurrence of data symbols, the shorter the entropy coding should be. Conversely, the smaller the probability of occurrence of data symbols, the longer the entropy coding, so that its average entropy coding is the shortest, if there are several pre- For the processing method, we should choose a method with the shortest average entropy coding length. Data corrupted by Rice must be packaged at the link layer. In order to adapt to the error-resistant design of space-oriented applications, a certain image range must be separated in the design of the algorithm to perform a truncation of prediction correlation. This truncation is to Rice. The packet is divided into the form, and the same compression scheme is adopted in the same Rice packet, that is, by predicting the correlation, eliminating the redundancy of the correlation between the data, and then compressing the de-correlated data. Therefore, the concept of image strips (distance between two reference samples) is introduced in the design of the scheme. The image strips can be a row of pixel values or a part of the entire map. The compressed stream of the image strip includes the compression identifier. (including K value), reference sample, sample compressed code stream and other parts. Inside the image strip, pixels can use each other in the strip to predict and decorrelate each other, but the pixel values of adjacent strips cannot be used to participate in the prediction calculation. This can limit the error loss to the packet and avoid errors. Code diffusion [5].

The encoding operation of 1.6RICE machine code can be divided into two stages of preprocessing and compression.

2.5.1 Pretreatment:

First, the source data should be statistically independent or unmemorized. Secondly, the n source data words are monotonically decremented by probability, ie.

P (i) is the probability of a data word with a data value of i

If it does not, the data word should be re-marked to achieve the monotonous decreasing requirement of the probability distribution. In this case, a re-marked lookup table needs to be transmitted.

2.5.2 Compression

(1) Three basic operations related to each other

1 basic sequence

For the data word xi , define the operator f s (xi)

$$f_s(x_i) = \frac{00.001}{x_i} \tag{1}$$

For data strings $x_1 x_2 \ldots x_n$, define the basic sequence

$$F_1(x) = f_s(x_1) * f_s(x_2) * \ldots * f_s(x_i) * \ldots * f_s(x_n) \tag{2}$$

Where * means that the bit pattern is linked, and the bit length of the basic sequence is

$$L(F_1(x)) = \sum_{i=1}^{n} L(f_s(x_i)) = n + \sum_{i=1}^{n} x_i \quad (3)$$

3. Coding basic sequence

Examine the basic sequence F1 (x) , which is a binary sequence consisting of 0 and 1 , defines the operator Ex tn (x) , and bases the sequence

F1(x) is divided into groups of equal length, each group of n bits, generally the last group is less than n bits, filled with 0.

Usually n= 3, so define 8 non-renewal codes

Input bit string	Output bit string
000	0
001	100
010	101
100	110
011	11100
101	11101
110	11110
111	111111

Defining the coding base sequence

$$F_2(x) = E_x t^3(F_1(x)) \quad (4)$$

4. Coding complementary basic sequence

Defined $F_1(x)$ as $F_3(x)$ a binary inverse, then you can define a coded complementary base sequence

$F_3(x) = E_x t^3(F_1(x))$

(2) Sele action of compression scheme

The principle is simple: use the scheme that produces the shortest code length. To avoid the compression ratio less than 1, define the fourth scheme as "do nothing"! Then there are the following seven seed algorithms:

F_0: Don't do anything

F_1: Mapping input data blocks into basic sequences

F_2: Encoding basic sequence

F_3: Encoding complementary basic sequences

F_4: Using the minimum value in the block as the predicted value, forming an interpolation sequence and generating a basic sequence based on the difference sequence

F_5: Encoding basic sequences based on differences

F_6: Encoding complementary base sequences based on differences

Each data block adds 3 bit overhead, indicating the selected sub-algorithm.

000 001 010 011 100 101 110

F_0 F_1 F_2 F_3 F_4 F_5 F_6

According to RICE recommendation, the input data block length J is selected from

$16 < j < 25(bite)$

The actual selection range can also be wider.

(3) Selection rules of sub-algorithms

Pair of F_i sub-algorithms D_i

$D0 = 8 \times J$

$D1 = L(FS)$

$D2 = INT(L(FS)/3) + 2J$

$D3 = INT(L(FS)/3) + 2(L(FS0 - J)$

$D4 = L(FSD) + 8$

$D5 = INT(FSD)/3 + 2J = 8$

$D6 = INT(L(FSD)/3) + 2(L(FS) - J) + 8$

Where:

J :block size

FS : basic sequence

FSD : basic sequence based on difference

$L(x) : x$: bit length

$INT(x) : x$: truncation

The smallest algorithm should be chosen D_i

(4) The composition of the compressed data stream

Sub-algorithm identification (3 bit)

Predicted value (for F4, F5, F6 sub-algorithms, 8 bit)

Compressed data (variable bit length)

In summary, the RICE algorithm divides the input data file into blocks, selects the optimal sub-algorithm, processes the data block, and forms a compressed data stream.

The RICE algorithm is divided into RICE-machine code and RICE-machine 2 code. The above describes the RICE-machine 2 code. In fact, if the F4, F5, and F6 sub-algorithms are removed, it is the RICE-machine code.

3 RIC E algorithm analysis

(1) If the original data is already a stationary memoryless source, the upper limit of the compression ratio for each data sample (all single-byte) is

$$CR_{\max} = 8/H \tag{5}$$

among them

$$H = -EP_j \log P_j \ (P_j \text{ is probability of } j \text{ occurrence of characters}) \tag{6}$$

The entropy of the source. At this point, there is no significant improvement in the RICE code compared to the Huffman code that has been optimized.

(2) A basic way to better compress such sources is to try to change the probability distribution of the source and jointly encode multiple data samples [6]. The RICE algorithm transforms the distribution of information sources by mapping the multi-symbol source into a binary bit string (or bit stream) by equation (1). But it is not difficult to see that if there are more "big values" in the data, this mapping will not be very effective, and even counterproductive! The RICE algorithm uses n=3 packet coding (actually Huffman code) for the mapped bit stream, trying to adopt the advantage of joint coding, but because the packet is too short, it is not necessarily superior to non-grouping such as arithmetic coding in principle. coding.

(3) If there is residual correlation between the data samples (strictly speaking, this does not meet the assumptions of the RIC E algorithm), then one-dimensional prediction is used to obtain certain benefits by introducing the RIC E-machine 2 code. Although this has a certain effect on the slowly changing data, since the minimum value within a block is simply subtracted instead of the true DPCM, the correlation between data cannot be effectively removed.

(4) If the source is non-stationary, the RICE algorithm can provide a seven-seed algorithm for selection, so it has certain adaptability. In principle, it has the potential to increase the compression ratio. This involves the choice of block length J. Obviously, reducing J will increase the adaptability, but it will also lead to an increase in overhead (≥3 bits per block), and in the worst case, even CR <1. Need to be compromised.

4.1 Conclusion

1.7.1 RICE's advantage over absolute arithmetic coding and Huffman coding is not absolute. Compared with the popular ARJ software, the RICE algorithm still has a gap; after de-correlation, the situation improves, but the RICE algorithm has no advantage over the other two entropy codes.

1.7.2 Since the sub-algorithm is simple and has no complicated operations, the real-time processing performance of the RICE algorithm cannot be ignored. All of its operations can be implemented at fixed points, which may be the reason why the algorithm was adopted by the European Space Agency and NASA [5].

1.7.3 Select different compression schemes according to different sources, with good compression effect and high real-time processing and adaptive statistics [7].

5. The introduction of LZW compression

5.1 Introduction to LZW compression algorithm

LZW compression is a compression method that compresses files into small files based on a table lookup algorithm. A special LZW compression algorithm uses a sequence of bits of a specified length (for example, 12 bits) and creates an entry for this particular bit pattern in a table (sometimes called a "dictionary" or "decodebook"). Strip and combine the pattern itself with the short code. As the input is read, any pattern that has been read will replace these short codes, effectively compressing the input into a smaller file. The decompressed decoder can create this table itself by using the algorithm as it was when encrypting the input [8].

Among them, the dictionary-based LZ series compression algorithm includes LZ77, LZ78 and LZW algorithms, among which LZW algorithm is developed on the basis of LZ77 and LZ78 algorithms. Because the compression effect of the LZW algorithm is better than the LZ77 and LZ78 algorithms, the LZW algorithm gradually replaces the LZ77 and LZ78 algorithms and is widely used in the processing of lossless compression of data.

As the total amount of aerospace telemetry data continues to increase, it exerts tremendous pressure on the transmission and storage of aerospace telemetry data channels. Therefore, the storage and transmission of aerospace telemetry data after compression is a reliable method to alleviate this phenomenon. The so-called data compression, also known as source coding, is to represent the signal sent by the source with a minimum of digital, reducing the signal space for a given message set or data collection set. The total amount of aerospace telemetry data is getting larger and larger. Therefore, it is necessary to perform lossless compression on aerospace telemetry data.

5.2 LZW algorithm example [9]

For example, the compression and decompression process of the LZW algorithm is assumed. The LZW algorithm needs to compress the input string sequence as "85, 65, 65, 65, 98, 105, 105, 105, 105, 95, 96", first initialize the dictionary to 256. The characters are 0~255, and then the input string sequence is compression-encoded. The encoding process is shown in Table 6. The decompression process is shown in Table 7. The dictionary can be dynamically reconstructed during the decompression process, so there is no need to transfer the dictionary during the transfer.

Table 1. LZW encoding process table

Current character	Input character	Dictionary entry	Output codeword
-	85		
85	65	Table(256)=(85, 65)	85
65	65	Table(257)=(65, 65)	65
65	65		
(65.65)	98	Table(258)=(65, 65, 98)	257
98	105	Table(259)=(98, 105)	98
105	105	Table(260)=(105, 105)	105
105	105		
(105, 105)	105	Table(261)=(105, 105, 105)	260
105	95	Table(262)=(105, 95)	105
95	96	Table(263)=(95, 96)	95
96	-		96

Table 2. LZW decompression process table

Input character	Current character I	Dictionary entry	Output codeword
85	-		85
65	85	Table(256)=(85, 65)	65
257	65	Table(257)=(65, 65)	(65, 65)
98	(65, 65)	Table(258)=(65, 65, 98)	98
105	98	Table(259)=(98, 105)	105
260	105	Table(260)=(105, 105)	(105, 105)
105	(105, 105)	Table(261)=(105, 105, 105)	105
95	105	Table(262)=(105, 95)	95
96	95	Table(263)=(95, 96)	96

The final decompressed output codeword is "85, 65, 65, 65, 98, 105, 105, 105, 105, 95, 96", which is identical to the input codeword sequence.

5.3 Advantages and disadvantages of LZW:

Since the space telemetry system has high requirements on the reliability and real-time performance of the system, the following three aspects should be considered when selecting the lossless compression algorithm:

(1) Compression ratio

$$CR = \frac{\text{Compressed file size}}{\text{File size before compression}} \tag{7}$$

Compression ratio, also known as Compression Ratio (CR), is the primary basis for measuring the compression effect of compression algorithms and can be calculated by equation (7). The file size before compression of CR file size (7) can be seen from the above equation. The smaller the compression ratio, the smaller the file size after compression, which means the better the compression effect. On the contrary, the larger the compression ratio, the higher the compression ratio. The worse the compression effect.

(2) compression decompression speed

Space telemetry data is transmitted in framing, and the data transmission rate is very high. Since the telemetry data takes a certain time in the stage of collecting framing, modulation and emission, etc., the speed of compression and decompression is high, and the compression and decompression time is required. Keep it as short as possible.

The corresponding compression and decompression times when the test data is compressed are shown in Table 3. Among them, each set of test data contains 512 bytes, and each group contains 1000 frames.

Table 3. Comparison of compression and decompression time of LZW compression algorithm

	Compression algorithm	Telemetry data1/s	Telemetry data2/s	Telemetry data3/s
Compression time	LZW	105.9272	429.3108	101.0617
Decompression time		12.4653	37.1092	11.2295

(3) Algorithm complexity

The compression and decompression algorithm needs to occupy certain hardware resources. Since the compression algorithm is implemented on the same FPGA as the capture, tracking and channel coding and decoding algorithms, and the FPGA resources are limited, the resources required to implement the compression algorithm are more Less is better.

The XZ7K325T-1 FPGA of Xilinx is used to implement the hardware implementation of the LZW algorithm. The FPGA resources occupied by the three algorithms are shown in Table 4.

Table 4. LZW compression algorithm occupies FPGA resource statistics

Resource Type	FPGA Total resources	LZW Occupied number
LUT	407600	73
Flip-Flop	203800	24
BRAM	445	16

The LZW algorithm uses variable length coding output, and the run length coding uses a run-length flag and a single-character mixed output. Theoretical analysis and simulation results verify that the inter-frame joint compression algorithm proposed in this paper is superior to the conventional lossless compression algorithm in compression ratio and compression time.

The advantages of LZW can be seen from the three aspects of compression rate, compression decompression speed and algorithm complexity. The compression ratio is lower than LZ77, the dictionary content does not need to be transmitted, the compression and decompression speed is faster, the algorithm complexity is moderate, and it is suitable for local correlation. Good data.

The shortcomings of LZW:

(1) The key frame is compressed by the LZW algorithm, and the inter-frame difference is compressed by using run-length coding. However, the premise of the inter-frame joint compression algorithm to obtain the compression effect is that there is a global correlation between the telemetry frames. If the telemetry frame data suddenly jumps, that is, the continuous multi-frame telemetry data has a large difference, after the run-length encoding compresses the inter-frame difference, the compressed codeword may be longer than the input codeword length.

(2) The dictionary must be created in the LZW compression and decompression process. The basic idea of the LZW dictionary multi-character parallel search method is to use a multi-tree tree to construct a dictionary, and use a multi-character parallel method to globally search the dictionary. The improved search method can speed up the dictionary search speed and shorten the compression delay without affecting the compression ratio, but the compression delay is still significantly longer than the traditional parallel search method or hash table search mode. In order to further shorten the compression delay, how to improve the construction and search of the dictionary will remain the focus of the next step.

6. The Huffman coding part

Huffman coding is a coding method proposed by DAHuffman in 1952. It is based on the frequency of occurrence of characters to construct the heterogeneous header code with the shortest average length. Sometimes it is called the best code. It is generally called Ha. Fuman coding.

6.1 Huffman coding principle [10]

3.1.1 The more frequently the probability of occurrence of a symbol, the shorter the codeword that is implemented by the Huffman coding principle, the lower the probability of occurrence of the symbol, and the longer the codeword being allocated.

2.1.2 The two symbols with the least frequency will have the same length of codewords, and they only have the least significant bits.

The average length of these codes is close to the entropy of the source.

6.2 The steps to implement the Huffman coding principle are as follows:

(1) Arranging the probability of occurrence of the source symbols in decreasing order;

(2) Combine the two minimum probabilities and continue this step, always placing the higher probability branch on top until the probability reaches 1.0;

(3) Specify 1 for the upper side of each pair and 0 for the lower one (or vice versa);

(4) Draw the path of each source symbol probability to 1.0, and record the 1 and 0 along the path;

(5) For each source symbol, a sequence of 1, 0 is written, and a Huffman code is obtained from right to left.

6.3 Algorithm and Implementation [11]

The r-ary encoding process of this algorithm is implemented as follows:

(1) Arrange q sources according to the probability distribution size in descending order: $p1 \geq p2 \geq ... \geq pQ$;

(2) Using 0, 1, 2, ⋯, r-1 code symbols to represent the r source symbols with the least probability, and merge the source symbols with the smallest r probability into one, so that the column contains only q-A new source S 1 of r+1 symbols is called a reduced source;

(3) The symbols of the reduced source S 1 are still arranged in descending order according to the probability size, and the symbols with the last r probability are merged into one symbol, and respectively, 0, 1, 2, ⋯, r-1 codes are used. The symbol indicates that a reduced source S 2 of q-2(r-1) symbols is formed;

(4) Continue in sequence until the source has only r symbols left. The last r source symbols are represented by r-ary symbols 0, 1, ⋯, r-1, respectively;

(5) Then, starting from the last stage of reducing the source and returning forward, the sequence of code symbols corresponding to each source symbol, that is, the corresponding code word, is obtained.

For r-ary Huffman coding, in order to make full use of short codes and make Huffman's average code length the shortest, the last reduced source must have r source symbols. Therefore, the number of source S symbols n1 needs to be supplemented by K probabilities. The sign of 0 needs to satisfy n1+K =S(r-1)+r, where S is the number of times the source is reduced. In MATLAB, the number of reductions S and the number of zeros can be implemented by the following statements:

```
N1=length(p);
s=ceil((n1-r)/(r-1));
k=(r-1)*s+r-n1;
```

The "m matrix" in the program records the symbol probability ordering label of the new source after each reduction. The Sort function implementation arranges the symbol probabilities in ascending order. In the program, the pre-r term probabilities of q[ ] are added together, and r-1 1s are added at the same time, and the complement 1 is not given a label. The generation process of the m matrix and the q matrix is described as follows:

```
For i=1:s+1
[q,l]=sort(q);
m (i,:)=[l(1:n-(r-1)*(i-1)),zeros(1,
(r-1)*(i-1))];
f=[q(1:r)];
x=sum (f);
q=[x,q(r+1:n),ones(1,
R-1)];
End
```

The "c matrix" in the program records the entire process of encoding, starting with the last line. For example, if the initial c matrix is set to n-i (the number of times of reduction), each line of n*n space characters is used to store n codewords, and each codeword has n symbols.

The core part of the encoding is implemented with the following three loop statements:

```
For i=2:s+1
c(s+2-i,1:n-1)=c(s+2-i+1,n*(find(m (s+2-i+1,:)==1))-(n -2): n*(find(m (s+2-i+1,:)==1)));
c(s+2-i,n)= '0'+r-1;
For w =2:r
c(s+2-i, (w -1)*n+1:w *n-1)=c(s+2-i,1:n-1);
c(s+2-i, w *n)='0'+r-w ;
End
For j=1:(r-1)*(i-1)
c(s+2-i,(j+r-1)*n+1:(j+r)*n)=c(s+2-i+1,n*(find(m(s+2-
i+1,:)==j+1)-1)+1:n*find(m(s+2-i+1,:)==j+1));
End
```

End

In the generation of the c-matrix, the elements that need to be sorted sequentially for each reduction can be completed with the third loop in the above code. The program returns the three return values of the function, encoding the final result h, the average codeword length l and the encoding efficiency yita.

3.4 Implementation results and performance analysis

In this paper, P=[0.2000 0.1900 0.1800 0.1700 0.1500 0.1000 0.0100]; as an example, the Huffman coding for different hexadecimals has the following results:

The above results further verify the two obvious features of Huffman code using probability matching method for source coding. Firstly, the encoding method of Huffman code ensures that the symbol with high probability corresponds to short code, and the symbol with small probability corresponds to long code. The short code is fully utilized; secondly, the last r codewords after each reduction are always the last one, so that the Huffman code is a compact instant code.

6.4 Conclusion

Although Huffman coding is more efficient than other coding methods in distortion-free coding, when the source data component is complex, the huge source set makes the Huffman code larger, the calculation amount of the code table generation increases, and the encoding code speed is rather slow; In addition, the unequal length coding makes the hardware decoding circuit difficult to implement, and the above reasons cause the practical application of Huffman coding to be limited. Second, the code obtained by the Huffman coding method is not unique, and the only reason is not as follows:

(1) Each time the source is reduced, the symbols 0, 1, ⋯, r-1 which give the last r probability of the source are the smallest. (For example, in this program, 0, 1, 2, ⋯r- 1 are assigned in turn)

(2) When the source is reduced, the probability that the symbols with the smallest r probabilities are combined is the same as the probability of other source symbols. When the two are probabilistically ordered in the reduced source, the position order may be arbitrary. In order to obtain the code with the smallest variance of the code length, the combined source symbol should be located at the highest possible position of the reduced source sequence, so that the short code can be fully utilized, and this is not considered in the implementation of this paper.

7. Conclusion

The goal of lossless image compression is to represent a given amount of data with as little data as possible without losing information. Predictive coding methods have always been the basic theoretical framework for image lossless compression. The various image lossless compression algorithms proposed so far are based on this idea. [12]

Different algorithms deal with different characteristics of data in terms of efficiency. Determine the compression effect of the algorithm, and consider a variety of indicators to consider. LZW+Huffman's algorithm is designed for the non-stationary nature of telemetry data. Although it is not the highest in compression ratio, its stability and real-time performance is very good, and the processed data can be forwarded in real time. Improve the modular design of hardware, high reliability, good versatility, easy to maintain and re-develop at the same time, real-time compression of telemetry data, and good application prospects after transmission through the network.

References

[1] Lin Wei, He Shuguang. Telemetry System for Manned Rocket. Missile and Space Vehicle Technology, 2006; 4: 5-10 North University, 2011

[2] Zhan Jianhua. Development of a lossless compression device for telemetry noise data based on DSP. Taiyuan

[3] Meng Nan, Yang Yanfei, Liu Wenyi Research on Real-time Lossless Compression Technology of Telemetry and Velocity Data. Vol. 13 No. 33 November 2013 <Science Technology and Engineering>
[4] Xu Zhili, Yan Jin. Application of Rice Lossless Data Compression Algorithm in Network Transmission
[5] Tang Weiqi, Wu Lenan, Zhan Xinnong. RICE algorithm compression effect measurement. Telemetry remote control Volume 19, Issue 3
[6] Wu Lenan. Principles and Applications of Data Compression. Electronic Industry Press, 1995
[7] Ren Yuli, Deng Jiaxian, Wu Wei, Wu Xiaoting. RICE-based lossless compression algorithm for remote sensing multispectral imagery. Communication Technology, Vol. 45, No. 09, 2012, P129-133
[8] Baidu Encyclopedia
[9] Liu Minli. Research and implementation of lossless compression technology for aerospace telemetry data [D]. Beijing Institute of Technology.2016.
[10] Wu Jiqun, Li Shuangke. Implementation of Huffman coding under MATLAB. China Science and Technology Information, No. 19, 2006
[11] Zhang Yuehua, Cui Wenchao. Implementation of MATLAB for R-element Huffman Coding. China Science and Technology Information 2008, No. 16
[12] Yu Jiabin, Yan Hainan, Yin Benyu. Design and Hardware Implementation of Telemetry Data Compression Algorithm. 2008, No. 12, Volume 41. Communication Technology