A LOSSLESS IMAGE COMPRESSION SYSTEM USING A BINARY ARITHMETIC CODER

Hua Ye, Guang Deng and John C. Devlin
Department of Electronic Engineering, La Trobe University
Bundoor, Victoria 3083, Australia
h.ye@ee.latrobe.edu.au
d.deng@ee.latrobe.edu.au
j.devlin@ee.latrobe.edu.au

Abstract

In this paper we incorporate a binary arithmetic coder into a predictive lossless image compression scheme. Its idea arises from the observation that a binary arithmetic coder needs much less computation than a multi-symbol one and therefore should run faster. To map the prediction errors to binary symbols which are fed into the arithmetic coder, Rice code has been adopted because of the Laplacian distribution for prediction errors. Experiments show that the compression performance of this scheme is better than that of LOCO and S+P and is close to that of CALIC.

1. Introduction

In most lossless predictive image compression algorithms, entropy coding is the essential tool to code the necessary information on the prediction and the prediction error [1]. Given a source alphabet $S = \{s_1, s_2, \cdots, s_n\}$ and the associated probabilities of occurrence $P = \{p(s_1), p(s_2), \cdots, p(s_n)\}$, the entropy coding problem is to find a mapping method for S so that the average code length for each symbol is minimised. Shannon [2] showed that for the best possible compression code, the average code length is the entropy of the source, i.e.,

$$H(S) = -\sum_{i=1}^{n} p(s_i) \log_2 p(s_i) \quad \text{bits/symbol.}$$

There are generally two types of entropy coding techniques: prefix codes and arithmetic codes. Prefix codes assign a single codeword to each symbol. The basic idea is to assign shorter codewords to more probable symbols and longer codewords to less probable symbols. Arithmetic codes, in comparison, assign a single codeword to the entire input sequence. To understand the fundamental principle of arithmetic coding, one can refer to [3]. Arithmetic codes almost always give better compression than prefix codes. Hence they are preferred in most leading lossless compression algorithms such as CALIC [4], LOCO [5] and S+P [6].

An arithmetic coder can be a multi-symbol coder or a binary one. In most image compression schemes, including those mentioned above, an arithmetic coder with a multi-symbol alphabet is used. This seems to be more natural because the prediction error always comes with multiple values. There are, however, some good reasons to use a binary coder:

1) The arithmetic coder has to maintain and update a probability table for all the possible symbols. If the coder is a binary one, there will be only a single probability in this table.
2) The arithmetic coder has to calculate the new intervals each time a symbol is input. For a binary coder, this calculation also becomes simpler, because there is only one endpoint change when a new bit is input.

These simplifications should make a binary coder very fast. Together with these attractive advantages, there are also some problems associated with image compression:

1) How to find an appropriate way of mapping the multiple values to distinct binary strings?
2) What is the influence of replacing a multi-symbol arithmetic coder with a binary one on the compression ratio? In other words, what is the cost of taking the above mentioned advantages?

In this paper, we propose a lossless compression scheme using a recently developed binary arithmetic coder, due to A. Moffat, *et al* [7]. Different symbol mapping methods are discussed, and then a Rice code based mapping strategy is proposed. To further exploit the residual redundancy in a neighbourhood of error points, a modification to the traditional raster scan order is also proposed. In Section 2, we briefly introduce our compression system. In Section 3, the proposed algorithm is applied to a set of test images and the compression results are compared with those from the three leading schemes. Finally, in Section 4, we present our conclusion.

0-7803-4325-5/98/$10.00

2. The Compression System

The block diagram of the proposed compression system is shown in Fig.1. Although the prediction and error quantisation blocks are also crucial to the final compression result, they are not the focus of this piece of work. We will only concentrate on the symbol mapping. We will also discuss the processing order.

2.1 The symbol mapping

There exist many methods of mapping a set of symbols to binary strings. Fixed length binary codes are the simplest choice. Obviously, they are not economical, since we can profit from the fact that some values occur more frequently than others. If shorter codes are assigned to more probable values, the average code length will surely be shorter than that of the fixed length binary code. The idea here is exactly the same as that of any prefix code. Thus we propose to use some kind of prefix code for the symbol mapping.

Huffman code [8] is by far the most famous and the best prefix code, but to construct a Huffman code book, the probability table for all the symbols must be available. This results in added computational expenses. A suboptimal but very easy to implement suffix code set was developed by Rice [9]. If the probability distribution of the symbols to be coded is exponential, it can be shown that a Rice code with the correct choice of its parameter produces an optimal prefix code. The distributions of the prediction errors for most images have long been considered to be close to a Laplace (or exponential) distribution. This suggests that Rice code be very suitable for image compression.

To encode an error value with Rice code, one has to map it to Rice numbers (0,1,2,...) first. Suppose these numbers have probabilities $p_0, p_1, p_2, \cdots$. These probabilities should satisfy:

$$p_0 \geq p_1 \geq p_2 \geq \ldots \quad (1)$$

There are normally two ways to do this:

1) The Rice number is the absolute value of the error. If it is not zero, code another bit for its sign.
2) Starting with 0, the positive and negative values are interlaced from low to high (0, +1, -1, +2, -2, ...). Then they are mapped to Rice numbers according to their order.

We already have the error quantisation block which maps the absolute value of an error value to a quantisation level number between 0 and 20. Since the probabilities of these level numbers satisfy equation (1), we can conveniently use these level numbers as the Rice numbers. This can be regarded as a method equivalent to 1).

Table 1 gives Rice codes for several values of the parameter k. This parameter k is referred to as the splitting factor because the Rice code can be generated in the following two steps:

1) The Rice number is divided into two parts: the k least significant bits (LSB's) which are normally non-compressible and the remaining most significant bits (MSB's).
2) The MSB part is coded into the fundamental sequence. (corresponding to column $k = 0$ in Table 1) Then this fundamental sequence is combined with the k LSB's.

Step 1) is actually a process of spliting the data value to be coded into two parts, one of which is non-compressible. This can also be regarded as accomplished in the error quantisation block because the quantisation error should be non-compressible. Therefore, $k = 0$ is a good choice for our scheme if we use the quantisation level number as the Rice number.

2.2 The processing sequence

Traditionally, the image pixels are processed in a raster scan order as shown in Fig. 2. Although it seems more straightforward and simpler to program, this order does not fully exploit the correlation within the immediate neighbourhood of each pixel when there is a change in row. Thus we propose a minor modification to it, as shown in Fig. 3. Experiments show that using this order can bring a roughly 3% decrease in the size of the compressed file.

1	2	3	4	5	6	7	8
9	10	11	12	13	14	15	16
17	18	19	20	21	22	23	24

Fig. 2 Normal order

1	2	3	4	5	6	7	8
16	15	14	13	12	11	10	9
17	18	19	20	21	22	23	24

Fig.3. The proposed order

2.3 The context modeling for entropy coding

Although we can assume that Rice code is suboptimal in our application, there still remains some correlation among the Rice codes for neighbouring points. For example, after a bit "0", it is highly probable for the next

bit to be "0". To remove such correlation, a context-based adaptive binary arithmetic coder is applied.

In a context-based arithmetic coder, there are m probability (frequency) tables if the total number of contexts is m. A context number c for the current pixel must be calculated using an appropriate scheme. Then the quantised prediction error for the current pixel is encoded with the c th probability table.

Our context modeling scheme shares the same idea with the CALIC, that is, both the structural information and the prediction errors in the immediate neiborhood are taken into account in the calculation of context numbers. In addition to the definition of the context numbers, we introduce a simple context filtering algorithm which leads to higher compression ratios. A detailed discussion on the context selection strategy is out of the scope of this paper. The interested reader can refer to [10].

3. Experiment Results

We applied the proposed compression algorithm to a set of commonly used test images. These images include human face, natural scenery, medical, radar and satellite, scientific and the "mandrill". The results are shown in Table 2. For comparison, results of CALIC, LOCO and S+P for the same set of images are also included in the table. It can be seen from this table that the average bit-rate for the proposed scheme is lower than those for LOCO and S+P. It is almost the same as that for CALIC.

4. Conclusion

A lossless image compression scheme is presented. This scheme uses Rice codes to map the quantised prediction errors to binary streams which then are coded with a context-based binary arithmetic coder. Because of the simplicity in Rice code formation and the lowest possible alphabet size for the arithmetic coding, this scheme has a very low computational complexity. Application of this compression scheme on a set of test images shows that the proposed algorithm has better compression performance than LOCO and S+P.

References

[1] N. D. Memon and K.Sayood, "Lossless image compression: a comparative study," in *Still image compression, SPIE-The International Society for Optical Engineering*, v.2418, pp.8-20, 1995.

[2] C. E. Shannon, "A mathematical theory of communication," *Bell System Technical Journal*, v.27, n.3, pp. 379-423, 1948.

[3] R. E. Blahut, *Principles and Practice of Information Theory*, Addison-Wesley, Reading, MA, 1987.

[4] X. L. Wu, *et al*, "A context-based, adaptive, lossless/near-lossless coding scheme for continuous-tone images," *ISO working document* ISO/IEC/SC29/WGI/N256, 1995.

[5] M. J. Weinberger, G Seroussi and G. Sapiro, "LOCO-I: A low complexity, context-based, lossless image compression algorithm," *Proc. Data Compression Conf.'96,* pp.141-150.

[6] A. Said and W. A. Pearlman, "An image multiresolution representation for lossless and lossy compression," *IEEE Trans. Image Processing*, vol. 5, pp. 1303-1310, 1996.

[7] A. Moffat, *et al*, "Arithmetic coding revisited," *Proc. Data Compression Conf.'95,* pp. 202-211.

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

[9] R. F. Rice, "Some practical universal noiseless coding techniques,"Jet Propulsion Laboratory, JPL Publication 79-22, Pasadena, California, 1979.

[10] G. Deng, "Symbol mapping and context filtering for lossless image compression," to be presented at *IEEE International Conference on Image Processing*, Chicago, Illinois, 1998.

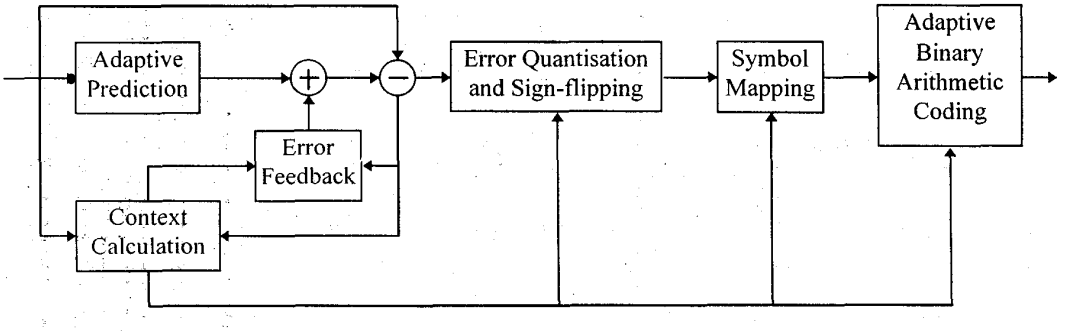

Fig. 1 Block diagram of the proposed compression system

Table 1 Rice codes for the first ten symbols.

Symbol	$k=0$	$k=1$	$k=2$	$k=3$
0	0	00	000	0000
1	10	01	001	0001
2	110	100	010	0010
3	1110	101	011	0011
4	11110	1100	1000	0100
5	$\underbrace{11\cdots}_{5}10$	1101	1001	0101
6	$\underbrace{11\cdots}_{6}10$	11100	1010	0110
7	$\underbrace{11\cdots}_{7}10$	11101	1011	0111
8	$\underbrace{11\cdots}_{8}10$	$\underbrace{11\cdots}_{4}100$	11000	10000
9	$\underbrace{11\cdots}_{9}10$	$\underbrace{11\cdots}_{4}101$	11001	10001
10	$\underbrace{11\cdots}_{10}10$	$\underbrace{11\cdots}_{5}100$	11010	10010

Table 2 The sizes (in byte) and the average bit-rates of the compressed test images

Image	**Proposed method**	**CALLIC**	**LOCO**	**S+P**
lenna	128500	129336	133289	131842
lena	134038	134615	139099	136643
woman1	149700	148817	153118	152510
woman2	103130	104761	108136	106233
CT	71764	70840	73067	81265
mri	26303	25828	27546	26297
xray	20729	21182	20183	22903
goldhill	151881	151682	154389	155508
Hursley house	144093	143953	144751	152946
F-16	115931	115976	118198	165673
lake	160259	160651	163273	165673
crowd	124146	123278	128275	131010
peppers	137512	137736	140538	143102
lax	185694	184358	188830	189939
airport	217681	214586	219972	217257
landsat	131003	130681	133667	133678
milkdrop	116670	116731	118968	121316
mandrill	189936	188022	193140	189719
Average bit-rate	4.1538	4.1439	4.2413	4.2845