A Simple Lossless Compression Heuristic for Grey Scale Images

L. Cinque[1], S. De Agostino[1], F. Liberati[1] and B. Westgeest[2]

[1] Computer Science Department
University "La Sapienza"
Via Salaria 113, 00198 Rome, Italy
e-mail: deagostino@di.uniroma1.it

[2] Computer Science Department
Armstrong Atlantic State University
Savannah, Georgia 31419, USA

Abstract. In this paper, we show a simple lossless compression heuristic for gray scale images. The main advantage of this approach is that it provides a highly parallelizable compressor and decompressor. In fact, it can be applied independently to each block of 8x8 pixels, achieving 80 percent of the compression obtained with LOCO-I (JPEG-LS), the current lossless standard in low-complexity applications. The compressed form of each block employs a header and a fixed length code, and the sequential implementations of the encoder and decoder are 50 to 60 percent faster than LOCO-I.

Keywords: grey scale image, lossless compression, differential coding, parallelization

1 Introduction

Lossless image compression is often realized by extending string compression methods to two-dimensional data. Standard lossless image compression methods extend model driven text compression [1], consisting of two distinct and independent phases: *modeling* [13] and *coding* [12]. In the coding phase, arithmetic encoders enable the best model driven compressors both for bi-level images (JBIG [7]) and for grey scale and color images (CALIC [18]), but they are often ruled out because too complex. The compression gap between simpler techniques and state of the art compressors can be significant. Storer [15] and Storer and Helfgott [16] extended dictionary text compression [14] to bi-level images to avoid arithmetic encoders, achieving 70 percent of the compression of JBIG1 on the CCITT bi-level image test set. Such method is suitable for high speed applications by means of a simple hashing scheme. A polylogarithmic time work-optimal parallel implementation of this method was also presented to further speed up the computation on the PRAM EREW [3, 4]. Such implementation requires more sophisticated architectures (pyramids or meshes of trees) [9] than a simple array of processors to be executed on a distributed memory system. The

extension of this method to grey scale and color images was left as an open problem, but it seems not feasible since the high cardinality of the alphabet causes an unpractical exponential blow-up of the hash table used in the implementation.

With grey scale and color images, the modeling phase consists of three components: the determination of the context of the next pixel, the prediction of the next pixel and a probabilistic model for the *prediction residual*, which is the value difference between the actual pixel and the predicted one. In the coding phase, the prediction residuals are encoded. A first step toward a good low complexity compression scheme was FELICS [8], which involves Golomb-Rice codes [6, 10] rather than arithmetic ones . With the same complexity level for compression (but with a 10 percent slower decompressor) LOCO-I [17] attains significantly better compression than FELICS, only a few percentage points of CALIC. As explained in section 2, also polylogarithmic time parallel implementations of FELICS and LOCO would require more sophisticated architectures than a simple array of processors.

The use of prediction residuals for grey scale and color image compression relies on the fact that most of the times there are minimal variations of color in the neighborood of one pixel. Therefore, differently than for bi-level images we should be able to implement an extremely local procedure which is able to achieve a satisfying degree of compression by working independently on different very small blocks. In this paper, we show such procedure. We present the heuristic for grey scale images, but it could be applied to color images by working on the different components [2]. The main advantage of this approach is that it provides a highly parallelizable compressor and decompressor. In fact, it can be applied independently to each block of 8x8 pixels, achieving 80 percent of the compression obtained with LOCO-I (JPEG-LS), the current lossless standard in low-complexity applications. The compressed form of each block employs a header and a fixed length code, and the sequential implementations of the encoder and decoder are 50 to 60 percent faster than LOCO-I. Two different techniques might be applied to compress the block. One is the simple idea of reducing the alphabet size by looking at the values occurring in the block. The other one is to encode the difference between the pixel value and the smallest one in the block. Observe that this second technique can be interpreted in terms of the model driven method, where the block is the context, the smallest value is the prediction and the fixed length code encodes the prediction residual. More precisely, since the code is fixed length the method can be seen as a two-dimensional extension of differential coding [5]. Differential coding, often applied to multimedia data compression, transmits the difference between a given signal sample and another sample.

In section 2, we sketch how FELICS and LOCO-I work and discuss their parallel complexity. In section 3, we explain our heuristic. Conclusions are given in section 4.

2 FELICS and LOCO-I

In this section, we sketch how FELICS and LOCO-I work and discuss their parallel complexity. As explained in the introduction, these are context-based method.

In FELICS the context for the current pixel P is determined by the last two pixels N_1 and N_2 read in a raster scan. If P is in the range defined by N_1 and N_2, then it is encoded with an adjusted binary code. Otherwise, the value distance from the closer of values N_1 and N_2 is encoded using a Golomb-Rice code. Golomb-Rice codes use a

positive integer parameter m to encode a non-negative integer n. Given the value m, $\lfloor n/m \rfloor$ is encoded in unary and n mod m in binary. A typical method of computing the parameter for Golomb-Rice code is to divide the image in 8x8 pixel blocks and select in a set of reasonable values the best one for each block with an exaustive search [11]. This method is not used in [8] because a better exaustive search is proposed for a sequential implementation. That is, to mantain for each context a cumulative total of the code length we would have at the current step for each reasonable parameter and to pick the best one each time.

Observe that the coding process would be highly parallelizable with the parameter selection of [11], since for each pixel of a block the parameter selection is completely independent from the rest of the image. A distributed memory system as simple as an array of processors would be able to perform the algorithm. With the other method instead, the design of a highly parallelized procedure becomes a much more complex issue, involving prefix computation and more sophisticated architectures than a simple array of processors [9]. However, with both methods the decoding process is hardly parallelizable, since to decode the current pixel the knowledge of the two previous pixels is required.

LOCO-I employs a more involved context modeling procedure where the context of pixel P in position (i, j) is determined by the pixels in positions $(i, j-1)$, $(i, j-2)$, $(i-1, j-1)$, $(i-1, j)$ and $(i-1, j+1)$. A count of prediction residuals and an accumulated sum of magnitudes of prediction residuals seen so far is maintained in order to select the parameter for the Golomb-Rice code. Moreover, in order to improve the compression performance a count of context occurences and an accumulated sum of prediction residuals encoded so far for each context also is maintained. These computations are used to reduce the prediction error. Further improvement is obtained by switching the Golomb-Rice code to encode characters from an extended alphabet when long runs of a character from the original alphabet are detected. All these operations require prefix computation in a highly parallelized version of the algorithm.

In conclusion, FELICS and LOCO methods do not provide highly parallelizable encoders and decoders implementable on a simple array of processors. The heuristic we present in the next section works independently on each 8x8 block of pixels. Since no information is shared among the blocks, a simple array of processors suffices to realize a constant time work-optimal parallel implementation.

3 A Simple Heuristic for Grey Scale Images

As previously mentioned, the heuristic applies independently to blocks of 8x8 pixels of the image. We can assume the heuristic reads the image with a raster scan on each block. The heuristic apply at most three different ways of compressing the block and chooses the best one. The first one is the following.

The smallest pixel value is computed on the block. The header consists of three fields of 1 bit, 3 bits and 8 bits repectively. The first bit is set to 1 to indicate that we compress a block of 64 pixels. This is because one of the three methods will partition the block in four sub-blocks of 16 pixels each and compress each of these smaller areas. The 3-bits field stores the minimum number of bits required to encode in binary the distance between the smallest pixel value and every other pixel value

255	255	255	254	254	110	110	110
255	255	255	254	254	110	110	110
255	255	255	254	254	110	110	110
255	255	255	254	254	110	110	110
255	255	254	128	127	128	129	130
255	253	253	128	128	129	130	131
254	253	252	129	129	130	131	132
253	252	251	130	130	130	254	255

Figure 1: An 8x8 pixel block of a grey scale image.

in the block. The 8-bits field stores the smallest pixel value. If the number of bits required to encode the distance, say k, is at most 5, then a code of fixed length k is used to encode the 64 pixels, by giving the difference between the pixel value and the smallest one in the block. To speed up the procedure, if k is less or equal to 2 the other techniques are not tried because we reach a satisfying compression ratio on the block. Otherwise, two more techniques are experimented on the block.

One technique is to detect all the different pixel values in the 8x8 block and create a reduced alphabet. Then, encode each pixel in the block using a fixed length code for this alphabet. The employment of this technique is declared by setting the 1-bit field to 1 and the 3-bits field to 110. Then, an additional three bits field stores the reduced alphabet size d with an adjusted binary code in the range $2 \leq d \leq 9$. The last componenent of the header is the alphabet itself, a concatenation of d bytes. Then, a code of fixed length $\lceil \log d \rceil$ bits is used to encode the 64 pixels.

The other technique compresses the four 4x4 pixel sub-blocks. The 1-bit field is set to 0. Four fields follow the flag bit, one for each 4x4 block. The two previous techniques are applied to the blocks and the best one is chosen. If the first technique is applied to a block, the corresponding field stores values from 0 to 7 rather than from 0 to 5 as for the 8x8 block. If such value is in between 0 and 6, the field stores three bits. Otherwise, the three bits (111) are followed by three more. This is because 111 is used to denote the application of the second technique to the block as well, which is less frequent to happen. In this case, the reduced alphabet size stored in this three additional bits has range from 2 to 7, it is encoded with an adjusted binary code from 000 to 101 and the alphabet follows. 110 denotes the application of the first technique with distances expressed in seven bits and 111 denotes that the block is not compressed. After the four fields, the compressed forms of the blocks follow, which are similar to the ones described for the 8x8 block. When the 8x8 block is not compressed, 111 follows the flag bit set to 1.

Image	OURS	LOCO
1	1.22	1.52
2	1.57	2.00
3	1.75	2.31
4	1.52	1.93
5	1.22	1.55
6	1.39	1.75
7	1.57	2.22
8	1.19	1.51
9	1.60	2.05
10	1.56	2.04
11	1.43	1.83
12	1.63	2.10
13	1.15	1.34
14	1.30	1.63
15	1.67	2.07
16	1.51	1.97
17	1.51	1.96
18	1.30	1.58
19	1.41	1.80
20	2.00	2.55
21	1.45	1.77
22	1.41	1.76
23	1.75	2.29
24	1.39	1.74
Avg	1.48	1.89

Figure 2: Compression ratios (uncompressed / compressed).

We now show how the heuristic works on the example of Figure 1.

Since the difference between 110, the smallest pixel value, and 255 requires a code with fixed length 8 and the number of different values in the 8x8 block is 12, the technique employed to compress the block is to work separately on the 4x4 sub-blocks. Each block will be encoded with a raster scan (row by row). The upper left block has 254 as smallest pixel value and 255 is the only other value. Therefore, after setting the 1-bit field to zero the corresponding field is set to 001. The compressed form after the header is 1110111011101110. The reduced alphabet technique is more expensive since the raw pixel values must be given. On the hand, the upper right block needs the reduced alphabet technique. In fact, one byte is required to express the difference between 110 and 254. Therefore, the corresponding field is set to 111000, which indicates that the reduced alphabet size is 2, and the sequence of two bytes 0110111011111110 follows. The compressed form after the header is 1000100010001000. The lower left block has 8 different values so we do not use the reduced alphabet technique since the alphabet size should be between 2 and 7. The smallest pixel value in the block is 128 and the largest difference is 127 with the pixel value 255. Since a code of fixed length 7 is required, the corresponding field is 111110. The compressed form after the header is (we introduce a space between pixel encodings in the text to make it more readable): 1111111 1111111 1111110 0000000 1111111 1111101 1111101 0000000 1111110 1111101 1111100 0000001 1111101 1111100 1111011 0000010. Observe that

Image	OURS	LOCO
1	33.8	64.7
2	32.9	60.5
3	32.2	59.5
4	33.0	60.6
5	34.1	67.4
6	32.2	60.7
7	34.9	60.1
8	34.7	67.0
9	32.7	58.6
10	35.3	60.0
11	33.2	62.5
12	34.4	60.3
13	34.0	68.2
14	33.9	64.5
15	32.3	60.5
16	33.3	60.5
17	33.3	60.0
18	33.4	64.9
19	33.2	61.9
20	25.5	48.1
21	32.8	61.1
22	33.4	62.8
23	33.2	56.6
24	33.2	63.8
Avg.	33.1	61.4

Figure 3: Compression times (ms.).

the compression of the block would have been the same if we had allowed the reduced alphabet size to grow up to 8. However, experimentally we found more advantageous to exclude this case in favor of the other technique. Our heuristic does not compress the lower right block since it has 8 different values and the difference between pixel values 127 and 255 requires 8 bits. Therefore, the corresponding field is 111111 and the uncompressed block follows.

We experimented this technique on the kodak image test set, which is an extension of the standard jpeg image test set. We reached 70 to 85 percent of LOCO-I compression ratio (Figure 2) (78 percent in average). The executions of our algorithm and LOCO were compared with a Intel Pentium 4, 2.00 GHz processor on a RedHat Linux platform. Our compression heuristic turned out to be about 50 percent faster (Figure 3). When we compared the decompression times, we obtain an even greater speedup (around 60 percent) in comparison with LOCO (Figure 4). This is not surprising since, as we mentioned in the introduction, while the LOCO compressor is more or less as fast as FELICS, the decompressor is 10 percent slower.

Conclusions

In this paper, we showed a simple lossless compression heuristic for grey scale images. The main advantage of this approach is that it provides a highly parallelizable compressor and decompressor since it can be applied independently to each block of 8x8

Image	OURS	LOCO
1	30.8	69.7
2	26.3	65.3
3	24.4	64.2
4	26.0	65.2
5	31.2	69.2
6	33.8	64.9
7	25.7	65.1
8	32.1	69.9
9	26.4	64.2
10	25.4	64.7
11	26.9	66.6
12	26.1	63.8
13	32.6	70.5
14	29.6	67.7
15	23.6	68.4
16	26.1	64.6
17	25.7	66.0
18	28.7	68.8
19	27.0	66.1
20	20.1	52.2
21	26.3	65.3
22	27.0	67.0
23	22.9	62.9
24	27.3	67.7
Avg.	27.2	65.8

Figure 4: Decompression times (ms.).

pixels. The compressed form of each block employs a header and a fixed length code. Two different techniques might be applied to compress the block. One is the simple idea of reducing the alphabet size by looking at the values occurring in the block. The other one is to encode the difference between the pixel value and the smallest one in the block. It was interesting to see that this technique achieves about 80 percent of the compression performance of LOCO-I and the compressor and decompressor are 50 to 60 percent faster. Also, our technique is definitely the easiest to implement and can be applied as well to color images.

References

[1] Bell T.C., Cleary J.G. and Witten I.H [1990]. *Text Compression*, Prentice Hall.

[2] Cinque L., De Agostino S. and Liberati F. [2004]. "A Simple Lossless Compression Heuristic for RGB Images", *IEEE Data Compression Conference*, 533.

[3] De Agostino S. [2002]. "A Work-Optimal Parallel Implementation of Lossless Image Compression by String Matching", *Proceedings Prague Stringology Club Conference*, 1-8.

[4] Cinque L., De Agostino S. and Liberati F. [2003]. "A Work-Optimal Parallel Implementation of Lossless Image Compression by String Matching", *Nordic Journal of Computing*, **10**, 13-20.

[5] Gibson J. D. [1980]. "Adaptive prediction in speech differential encoding system", *Proceedings of the IEEE*, **68**, 488-525.

[6] Golomb S. W. [1966]. "Run-Length Encodings", *IEEE Transactions on Information Theory* **12**, 399-401.

[7] Howard P. G., Kossentini F., Martinis B., Forchammer S., Rucklidge W. J. and Ono F. [1998]. "The Emerging JBIG2 Standard", *IEEE Transactions on Circuits and Systems for Video Technology*, **8**, 838-848.

[8] Howard P. G. and Vitter J. S. [1993]. "Fast and Efficient Lossles Image Compression", *IEEE Data Compression Conference*, 351-360.

[9] Leighton F. T. [1992]. *Introduction to Parallel Algorithms and Architectures*, Morgan-Kaufmann.

[10] Rice R. F. [1979]. "Some Practical Universal Noiseless Coding Technique - part I", *Technical Report JPL-79-22*, Jet Propulsion Laboratory, Pasadena, California, USA.

[11] Rice R. F. [1991]. "Some Practical Universal Noiseless Coding Technique - part III", *Technical Report JPL-91-3*, Jet Propulsion Laboratory, Pasadena, California, USA.

[12] Rissanen J. [1976]. "Generalized Kraft Inequality and Arithmetic Coding", *IBM Journal on Research and Development* **20**, 198-203.

[13] Rissanen J. and Langdon G. G. [1981]. "Universal Modeling and Coding", *IEEE Transactions on Information Theory* **27**, 12-23.

[14] Storer J.A. [1988]. *Data Compression: Methods and Theory* (Computer Science Press).

[15] Storer J. A.[1996] "Lossless Image Compression using Generalized LZ1-Type Methods", *IEEE Data Compression Conference*, 290-299.

[16] Storer J. A. and Helfgott H. [1997] "Lossless Image Compression by Block Matching", *The Computer Journal* **40**, 137-145.

[17] Wimberger M. J., Seroussi G and Sapiro G. [1996] "LOCO-I: A Low Complexity, Context Based, Lossless Image Compression Algorithm", *IEEE Data Compression Conference*, 140-149.

[18] Wu X. and Memon N. D. [1997] "Context-Based, Adaptive, Lossless Image Coding", *IEEE Transactions on Communications*, **45**, 437-444.